
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# MOBILNÍ APLIKACE PRO SPRÁVU A REZERVACI PRODUKTŮ

**MOBILE APPLICATION FOR MANAGEMENT AND RESERVATION OF PRODUCTS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **MAREK JOUKAL**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **JAKUB ŠPAŇHEL, Ing.**
**SUPERVISOR**



**BRNO 2017**

# Abstrakt

Bakalářská práce se zabývá návrhem a následně vývojem mobilní aplikace pro operační systém Android jako podpůrná aplikace zároveň vyvíjené webové služby rezervačního systému. Cílem práce je vytvořit vizuálně minimalistickou avšak intuitivní a především jednoduchou aplikaci, pomocí které by uživatel mohl efektivně spravovat svá zařízení či předměty ať už v domácnosti nebo v práci.



# Abstract

This bachelor thesis deals with the design and subsequent development of the mobile application for the Android operating system as a supporting application of the simultaneously developed web service of the reservation system. The goal of the thesis is to create a visually minimalistic yet intuitive and, above all, simple application, by which the user can efficiently manage his / her devices or objects, whether in home or at work.



# Klíčová slova

mobilní aplikace, správa předmětů, Android, design, uživatelské rozhraní, inventarizace

# Keywords

mobile applications, object management, Android, design, user interface, physical inventory.



# Citace

JOUKAL, Marek. *Mobilní aplikace pro správu a rezervaci produktů.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Jakub Špaňhel, Ing.

# Mobilní aplikace pro správu a rezervaci produktů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jakuba Špaňhela. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Marek Joukal
17. května 2017

## Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Jakubovi Špaňhelovi za odborné vedení, rady a věcné připomínky během vypracovávání této práce. Dále bych chtěl poděkovat všem lidem, kteří se podíleli na testování nebo jiné podpoře.

# Obsah

# Kapitola 1

# Úvod

Informační technologie a moderní doba nás obklopuje každý den a jen velmi těžko se můžeme těmto tlakům vyhnout. Tyto technologie stále více pronikají i do sféry správy a administrace malých i velkých firem. Dnešní doba klade stále větší požadavky na uchovávání dat, záznam informací a inventarizaci firemního majetku. To vede ke stále větší potřebě uživatelsky přívětivých aplikací, které se zabývají touto problematikou.

Ačkoliv se zvyšuje potřeba aplikací řešících inventarizaci a správu majetku, nabídka těchto programů je na trhu stále poměrně omezená. Mnoho stávajících systémů pro správu a zaznamenávání historie majetku je primárně určená pro podnikovou sféru a jejich náklady na pořízení a údržbu bývají značně vysoké. Z důvodu vysoké ceny jsou tato řešení dostupná spíše pro korporátní sféru. Většina systémů pro spravování majetku bývá součástí velkých informačních systémů, které jsou těžkopádné a jejich užívání je fixované k počítači. Tyto systémy z důvodu komplexnosti nemají mobilní adaptaci. To může být jedna z příčin, proč jiné segmenty žádnou takovou aplikaci nevyužívají a raději dávají přednost tužce a papíru.

Na trhu se nevyskytuje minimalistická aplikace, která by nebyla vázaná na komplexní informační systém. Ačkoliv tato konkrétní funkcionalita se vyskytuje pouze v systémech pro firemní správu, existují aplikace, které s jistou rezervou lze použít jako správce předmětů. Na druhou stranu tyto alternativy jsou značně nedostačující pro správu většího množství položek majetku.

Tyto nedostatky by mohla do značné míry vyřešit aplikace pro správu a rezervaci produktů. Hlavní předností aplikace by měla být jednoduchost a intuitivnost uživatelského prostředí. Aplikace by nabízela možnost vytvářet libovolně nástěnky, díky kterým by mohli uživatelé kategorizovat a třídit různé typy předmětů. Jeden z aspektů aplikace by také byla funkce vytvořit týmy umožňující snadnější správu nástěnek. Výhodou aplikace by měla být vysoká flexibilita oproti desktopové verzi. Uživatel by nebyl vázán k počítači, a tudíž by mohl podstatně rychleji interagovat se systémem a efektivněji provádět správu položek.

Cílem této práce je navrhnout a implementovat mobilní aplikaci, která se zaměřuje na správu předmětů a měla by být efektivní, intuitivní a snadno ovladatelná. Popisy postupů, jak těchto aspektů dosáhnout, se budou prolínat celou technickou zprávou, protože se jedná o stěžejní požadavky na aplikaci.

# Kapitola 2

# Analýza a současná situace na trhu

V této kapitole jsou rozebrané důležité pojmy, které doprovází několik realizovaných rozhodnutí, a budou tedy doprovázeny důvody, proč některé funkce byly implementovány právě zvolenou cestou. Zároveň jsou popsány požadavky na aplikaci a analyzován současný stav aplikací s podobnou funkcionalitou.

## 2.1 Intuitivita aplikace

Jedním z hlavních požadavků na aplikaci je intuitivní ovládání prostředí, které zároveň bude dostatečně rychlé bez zbytečných procedur při vykonávání základních úloh. Aby bylo možné realizovat takový požadavek, je potřeba od základu analyzovat jeho problémy a následně rozebrat jejich možná řešení.

Intuitivitu ovlivňují dva hlavní faktory a to rozvržení grafického rozhraní a design. Tyto faktory mají k sobě blízko, ovšem každý z nich také ovlivňuje i prvky, které druhý faktor neovlivňuje. Rád bych také zmínil, že názor na intuitivitu a design je z velké části subjektivní a je zde tedy velké spektrum názorů co může či nemusí být intuitivní.

Kterýkoliv uživatel dnešní technologie si může povšimnout ne pouze technologické, ale také designové evoluce. Ačkoliv se na první pohled mohou zdát pravidla designu jednoduchá, při bližším zkoumání lze zjistit, že tomu tak není, ba naopak můžeme sledovat, že tento obor je rok od roku čím dál více důležitější při vývoji programů, aplikací, her a dalších nástrojů.[13] Při vývoji různých produktů design plní svou funkci různou prioritou (vizualizace, animace, zvuk). V případě nástroje pro spravování a rezervování předmětů, je důležitá vizuální stránka a v pokročilejším stádiu může být i důležitá animace.

Za poslední roky bylo vytvořeno několik designových jazyků[1]. Tyto jazyky zastřešují nějaký styl nebo schéma a popisují jejich charakteristiku, čímž se vyznačují a čímž se posléze stávají ikonickými. Návrháři takového designu kompletují sadu pravidel pro charakteristický ale zároveň konzistentní vzhled. Tato pravidla nejsou zcela striktní, popisují různé možnosti, jak lze různé elementy vizuálně zpracovat a implementovat do vyvíjené aplikace. Tato pravidla se mohou týkat tvarů, barevných schémat, vzorů, textur nebo dokonce i rozvržení. Ačkoliv pravidla nejsou nijak striktní, čím více se bude vývojář aplikace rozcházet s dokumentací designu, tím méně může aplikace připomínat vybraný styl, zvláště pak pokud se vývojář rozhodne rozcházet i s elementárními resp. ikonickými pravidly. Tyto designové jazyky se pak mohou shlukovat do větších obecnějších popisů (žánr) nebo nao-

[1] Anglicky *Design Language*. v literatuře se může vyskytovat Guidelines - konkrétní dokuemntace designového jazyka

pak dělit na různé variace stylu. Tyto jazyky se zpravidla snaží o konzistentní vzhled vícero aplikací, přehledného rozhraní a intuitivní rozvržení ovládacích prvků. Kromě toho mají za cíl usnadnit práci vývojářům, aby nemuseli vytvářet tato pravidla zcela od základu sami a mohli se držet již sepsaných pravidel.

Posledním trendem těchto jazyků je žánr tzv. Plochý design (*Flat Design*) z něhož vychází *Material Design* od společnosti Google, Inc. a *Modern UI* (dříve *Metro*) od společnosti Microsoft Corporation. Tento žánrový trend lze pozorovat i u jiných společností a i menších vývojářů využívajících tyto jazyky. Tento trend se vyskytuje především u systémových rozhraní, webových aplikací, aplikací pro chytré telefony, a dalších. Na druhou stranu například ve videoherním designovém segmentu není nástup trendu tak zřejmý, jako u předchozích vyjmenovaných. *Modern UI* se díky společnosti Microsoft může těšit z masivního rozšíření pomocí operačního systému Windows. Tak stejně lze hovořit o *Material Designu*, který Google používá jak ve svých aplikacích, tak na svém mobilním operačním systému Android. Výskyt *Modern UI*, který se uplatňuje na všech produktech společnosti Microsoft od vydání verze operačního systému Windows 8, je značně vysoký. Jen u desktopového operačního systému tvoří téměř 50% uživatelů.[10] U operačního systému Android je situace ještě podstatně markantnější. V mobilní sféře se podíl Androidu a tudíž i *Material Designu* pohybuje okolo 72% uživatelů.[11] V globálním měřítku včetně desktopových Windows se k březnu 2017 Android umístil na pozici globálně nejpoužívanějšího operačního systému.[12] Na základě rozšířenosti Plochého designu se dá odvodit, že většina uživatelů je na taková prostředí navyklá, protože se s nimi dostávají denně do kontaktu.

## 2.2 Material Design

Jak již bylo zmíněno, součástí designového jazyka může být, kromě barevných palet, tvarů atp., i rozvržení elementů. Tím rozumějme tlačítka (jejich pozice, odsazení, velikost, a další), nadpisy, obrázky, textová pole a jiné. Tyto jazyky mívají veřejně dostupnou dokumentaci, aby se jí mohli řídit další vývojáři.

V rámci mé práce jsem se rozhodl pro využití dokumentace designového jazyka *Material Design*. Důvodů pro toto rozhodnutí je hned několik. Jeden z hlavních je skutečnost, že Android i *Material Design* pochází od společnosti Google, která se pochopitelně snaží na této platformě rozšířit tento styl. Z toho logicky vyplývá, že uživatelé platformy Android znají styl *Material Design* a nebude jim cizí při používání i mé aplikace. Avšak existují i různé doplňky na platformu Android, které stylují prostředí telefonu na velmi podobné jiné platformě, jako například Windows 10 (mobilní adaptace) resp. *Modern UI*.

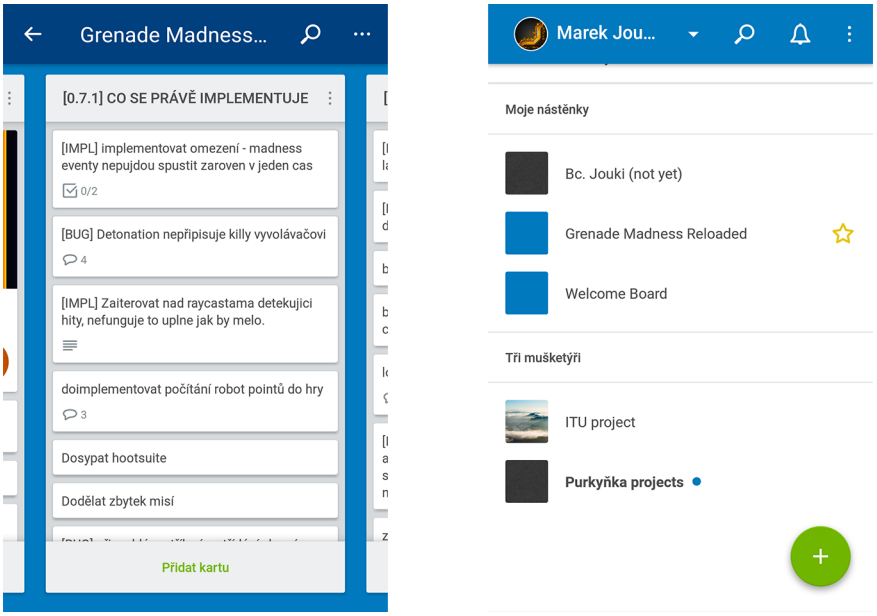
*Material Design* se snaží o přehlednost prvků. Toho dosahuje tím, že mezi jednotlivé elementy přidává značné odsazení. Barevná paleta je decentnější avšak pestrá, zvláště pokud se bavíme o prostředí aplikace (existují motivy tapet, které jsou velmi barevně kontrastní). Ikony jsou rovněž ploché, tzn. jsou vynechané detaily a vyobrazeny pouze základní tvary. Velmi často lze vidět i monochromatické ikony namísto barevné varianty. *Material Design* obsahuje i dokumentaci textu jak by měla aplikace komunikovat s uživatelem. Tuto část jsem neimplementoval z důvodů absence lokalizace do češtiny a tudíž neoficiálního pokusu o překlad těchto frází. Mojí prioritou bylo rozvržení jednotlivých obrazovek aplikace.[6]

## 2.3 Současná situace

Současná situace na poli rezervačních systémů či udržování informací o tom, kdo si vypůjčil jaký předmět, s udržováním záznamů, může mít hned několik přístupů, které záleží na měřítku specifičnosti. V případě velmi obecného zaměření na základní funkcionality aplikace se nabízí hned několik možností. Apliakce pro správu TODO listů sice lze použít jako provizorní alternativu, ovšem taková varianta má pochopitelně své nedostatky, protože není dedikovaná pro tyto účely. Čím více podmínek na aplikaci budeme klást, tím více se bude zužovat výběr těchto aplikací. Takovéto aplikace jsou do jisté míry podobné aplikacím pro správu projektů, kde je možné vytvořit různé úkolové tabule, vytvářet seznamy s možností si odškrtnout dokončené podúkoly celku, přiřazovat je jednotlivým členům a mnoho dalších podobných funkcí. Tato komplexnost může paradoxně mít nežádoucí efekt zahlcení přebytečnými funkcionalitami. Pro správu předmětů většina těchto funkcí není potřeba. Tím se postupně dostáváme k důvodům, proč vznikla moje aplikace.

Velké množství aplikací umožňuje například pouze odškrtnutí dokončeného úkolu, nebo mají nevyhovující strukturu, a tudíž přesouvání položek z listu do jiného listu je zdlouhavé, nebo zbytečně obtížné, jelikož aplikace není na tento způsob využívání přizpůsobena. Jednou z aplikací, která splňovala většinu bodů specifikace (viz kapitola 2.4) pro cílovou aplikaci, je Trello od společnosti Trello Inc.[2] (viz obrázek 2.1). Z tohoto důvodu se také Trello stalo v určitých ohledech inspirací pro vytvoření funkčního prototypu, co se týče návrhové části.

Kromě mobilní platformy současně vzniká i webová varianta tohoto projektu.




Obrázek 2.1: Mobilní aplikace Trello, která byla v ranné fázi významnou inspirací pro rozvržení prvků a způsob interakce s aplikací.

[2] https://trello.com/

## 2.4 Nativní aplikace a dedikovaná funkcionalita

V současné době vlastní chytrý telefon 80% globální online populace.[14] V lokálním měřítku České republiky má chytrý telefon 58% populace.[15] Oproti loňskému roku jde o roční nárůst 3%.[1] Jednou z hlavních výhod mobilního zařízení je samozřejmě přenosnost a kompaktnost. Díky těmto, a nejen těmto, vlastnostem se nativní aplikace v mobilu stávají populárnější než webová adaptace. To dodává i několik dalších výhod, které mohou být využity:

- Možnost využití technologií, které v prohlížeči dostupné nejsou.
- Efektivnější distribuce aplikace (a jejich aktualizací) pomocí centralizovaných obchodů.
- Záruka stejného vzhledu aplikace na všech zařízeních platformy i přes různá rozlišení.
- Možnost implementace lepších/intuitivnějších ovládacích prvků.
- Možnost přizpůsobit aplikaci přímo na některé typy zařízení, díky čemuž lze dosáhnout optimalizace aplikace.
- Rychlejší přístup k žádaným datům.

Jak již lze usoudit z vypsaných důvodů, nativní aplikace nese několik výhod. Na druhou stranu by se dalo vyjmenovat i několik nevýhod pro nativní aplikace oproti webové platformě (jako například oddělený vývoj od webové platformy), nicméně její výhody převyšují – především z důvodu pohodlnějšího přístupu k informacím na mobilu.

Dalším aspektem nativní aplikace je bezpochyby vzhled. Využití *Material Designu* nahrává jeho využití, jelikož dokumentace je psaná prioritně pro nativní aplikace – různé hodnoty velikosti prvků, odsazení atp. jsou například psané v jednotce dp (Density-independent Pixels), která se u mobilních zařízení primárně používá a můžeme jí rozumět jako univerzální jednotkou na rozdíl od klasických hodnot pixelů, kde velikost elementu je přímo závislá na rozlišení zobrazovacího panelu.

## 2.5 Analýza problematiky

Základní struktura aplikace byla stanovena tak, aby se shodovala s webovou adaptací (dále jen web), tzn., že uživatelé mohou být členy nástěnek a členy týmu. Jednotliví uživatelé mohou být součástí vícero týmů a vícero nástěnek. Tyto týmy pak mohou být přiřazeny jako skupina uživatelů k nástěnce, a tím získat přistup k nástěnce. Uživatel může mít mnohonásobný přístup k jedné nástěnce a to jak přes vícero týmů, které jsou k této konkrétní nástěnce přiřazeny, ale také jako samotný uživatel. Uživatel by neměl ztratit přístup, pokud má alespoň jeden z těchto přístupů stále k dispozici, tedy přes alespoň jeden tým, kterým je členem, nebo jako samotný uživatel (vůči nástěnce). Při odstranění uživatele ze všech možných přístupů k nástěnce je potřeba se postarat o jeho vypůjčené předměty. Z důvodu kompatibility s webem je dále potřeba nastavovat další vlastnosti (jako například url), ale převážnou většinu nastavuje sama aplikace, ačkoliv je sama nevyužívá, například soukromí nástěnky. V aplikaci je sice možnost nastavit její soukromí, ale z důvodu přehlednosti, způsobu využívání aplikace a možností zahlcení přebytečnými daty, je tato funkce implementována pouze v rámci kompatibility. Všechny nástěnky se chovají jako v soukromém

nastavení. To znamená, že v aplikaci není seznam veškerých vytvořených veřejných nástěnek serveru a členem se tedy lze stát pouze přidáním již stávajícím členem. Logika týmu je téměř identická.

## 2.6 Analýza požadavků

Na rozdíl od webové adaptace se u mobilní verze čeká minimalistické ovládání pro základní zacházení. Na základě odpovědí od několika dotazovaných subjektů jsem usoudil, že velkému množství dotazovaných dělá problém především způsob záznamu o vypůjčení předmětu, ale naopak by velká část uvítala možnost přesunu na mobilu. Dotazník, který byl použit k průzkumu je k nahlédnutí v příloze A. Z toho tedy lze vyvodit, že je potřeba základní operaci přemisťování předmětů, udělat co nejrychleji dostupnou. Většina respondentů se dělí na dvě skupiny, podle toho, jakým způsobem by aplikaci využívali. První skupina by vypůjčené předměty, zaznamenávala na konci pracovní směny, protože si vypůjčený předmět chtějí odnést domů. Zatímco druhá skupina předměty využívá přímo na svém pracovišti a tudíž by předměty zaznamenávala během pracovní doby. Po prvotní analýze bylo zřejmé, že přesouvání předmětů bude nejspíše nejčastější prováděná akce.

Podstatná množina respondentů, kteří odpověděli kladně na otázku, zda již používají nějakou aplikaci pro vytváření TODO listů, také odpověděli, že používají již jmenované Trello, což umocňuje opodstatnění, že aplikace může být pro rannou fázi vývoje značnou inspirací. Zároveň si od tohoto postupu slibuji, že mohu předejít některým nedostatkům, které by mohli vzniknout, pokud bych celý návrh vytvářel od základu.

Vzhledem k paralelně vyvíjené webové adaptaci bylo nutné dohodnout společnou strukturu a logiku dat, z tohoto důvodu byla část analýzy předem určená, aby nedocházelo k rozcházejícímu se chování aplikace.

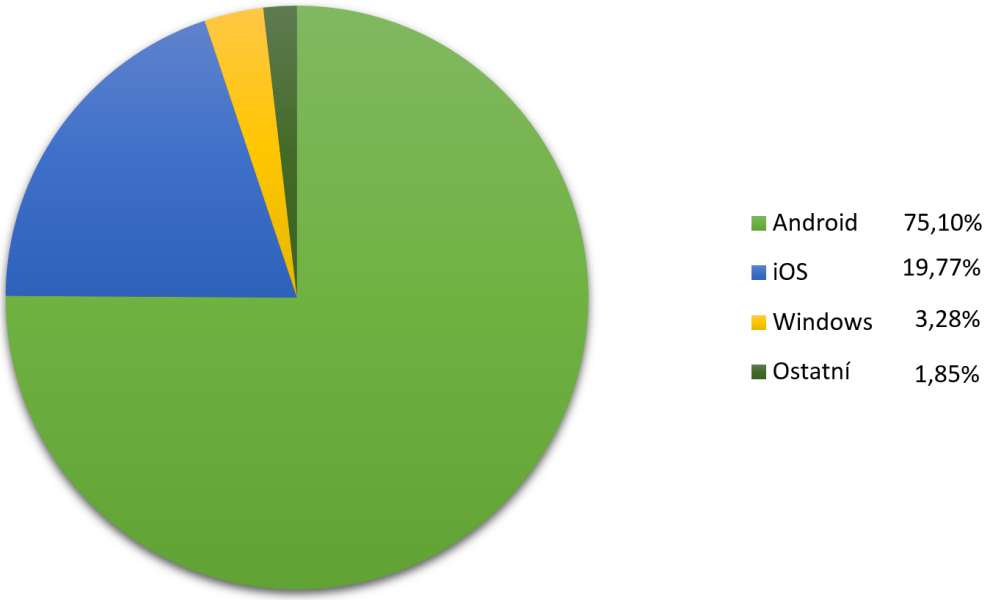
## 2.7 Výběr mobilní platformy

Na samotném začátku, ještě před analýzou, bylo potřeba rozhodnout, pro jaké platformy bude aplikace vyvíjena. Bylo potřeba zjistit náročnost vývoje na jednotlivé mobilní operační systémy a také možnosti složeného vývoje na vícero platforem zároveň. Na základě časového omezení, složitosti a zkušeností s takovýmto vývojem bylo rozhodnuto, že aplikace bude vyvíjena pouze na jednu platformu. Následně bylo tedy třeba rozhodnout, která platforma bude pro tento projekt nejvhodnější. Vybraná platforma měla splňovat, pokud možno, co nejvíce stanovených kritérií:

- Platforma musí být značně rozšířena
- Platforma nesmí být zastaralá a vývoj na ní by neměl být kontraproduktivní
- Budoucnost platformy by neměla být nejistá
- Možnost případné distribuce aplikace do online obchodu a zároveň bez velkých nákladů

Po vyhodnocení grafu poměru rozšíření jednotlivých platforem (viz obrázek 2.2) a zhodnocení ostatních platforem, byla jako optimální zvolena platforma Android. Ačkoliv iOS je druhý nejrozšířenější systém, jeho procentuální zastoupení, spolu s ostatními, je podstatně nižší než dominantní Android. Navíc případná distribuce v oficiálním obchodě by

obnášela značné náklady. Podobná situace platí i u Windows (Phone). Ačkoliv byl systém dříve velmi rozšířeným, dnes, jak můžeme vidět, tomu tak již není a tím pádem nesplňuje podmínky ohledně kontraproduktivity a nejisté budoucnosti platformy. Zbývající systémy jsou buď značně zastaralé, nebo jejich podíl je kriticky nízký. Dalším důvodem může být i omezená možnost testování na těchto platformách. Naprostá většina z potencionálních uživatelů vlastní mobilní telefon právě s operačním systémem Android a jen ve výjimečných případech s platformou iOS. V případě vývoje iOS a dalších platforem by byl problém s pozdějším testováním a získávání zpětné vazby, což by v případě systému Android podle mého předběžného průzkumu v mém okolí, neměl být problém.




Obrázek 2.2: Podíl jednotlivých mobilních platforem v České republice k dubnu 2017.

### 2.7.1 Výběr verze API

Ještě před samotným začátkem návrhu aplikace, bylo třeba určit, jaké minimální Android API bude aplikace podporovat. Jeden z aspektů vybrané verze API je následná práce s implementací grafického návrhu. Starší verze API nepodporují mnoho moderních grafických prvků, které se dají alespoň částečně kompenzovat podpůrnými knihovnami, byť jen omezeně. Pro úplnou kompatibilitu by bylo potřeba větvit aplikaci na několik verzí dle verze systému, což je velmi neefektivní. V tabulce 2.1 můžeme vidět, že většina uživatelů mají nainstalovanou verzi 4.0 až 7.0.

Z důvodu co nejširší kompatibility jsem se rozhodl aplikaci vyvíjet pro Android 4.0.3 Ice Cream Sandwitch API verze 15. Výběrem verze 15 bude pokryto okolo 99% všech zařízení. Toto rozhodnutí ovšem nebrání používat nejnovější prvky z nové verze API. Jak už bylo zmíněno, tuto úlohu zařizují podpůrné knihovny *Android Support Library*. Ty zajišťují dopřednou kompatibilitu. V opačném případě by tyto nové prvky nemohly fungovat.

| **Verze** | **Název** | **Verze API** | **Podíl** |
|---|---|---|---|
| 2.3.3 | Gingerbread | 10 | 0,9% |
| 4.0.3 - 4.0.4 | Ice Cream Sandwitch | 15 | 0,9% |
| 4.1.x | Jelly Bean | 16 | 3,5% |
| 4.2.x | | 17 | 5,1% |
| 4.3 | | 18 | 1,5% |
| 4.4 | KitKat | 19 | 20,0% |
| 5.0 | Lollipop | 21 | 9,0% |
| 5.1 | | 22 | 23,0% |
| 6.0 | Marshmallow | 23 | 31,2% |
| 7.0 | Nougat | 24 | 4,5% |
| 7.1 | | 25 | 0,4% |

Tabulka 2.1: Tabulka ukazující podíl jednotlivých verzí platformy Android (a jejich verzi API). Data jsou platná k dubnu 2017. (v tabulce nejsou uvedené verze, které mají nižší podíl než 0,1%)

# Kapitola 3

# Návrh mobilní aplikace

V této kapitole je detailněji popsán návrh grafického prostředí, procesy změn v návrhu, které vznikly na základě zpětné vazby a kde proběhly změny. Také zdůvodním vybrané způsoby realizace v oblastech, kde existovalo několik metod, jak daný problém řešit.

## 3.1 Uživatelské rozhraní

Uživatelské rozhraní je pro aplikaci stěžejní částí, a proto bylo třeba udělat několik důležitých rozhodnutí, jako například: jak bude aplikace strukturovaná, jakou barevnou paletu bude využívat či odkud budou dostupné jednotlivé funkce programu.

### 3.1.1 Návrh aktivit

Při vytváření návrhu aktivit je potřeba si určit základní strukturu aplikace. Je nutné určit, kde se budou vyskytovat jaké obrazovky aplikace (dále také aktivita), jakou budou mít funkcionalitu a taky, jak bude možné se zanořovat z jedné aktivity do aktivity druhé. Kromě zanořování je i potřeba rozhodnout, které aktivity by měly být přístupné v globálním měřítku, zatímco ostatní budou dostupné pouze kontextově (na základě toho kde se uživatel právě v aplikaci nachází).

U prvotního návrhu jsem si dovolil do jisté míry experimentovat s jiným, nekonvenčním, ovládáním, než je zavedeno u jiných aplikací, s pokusem najít různé alternativy ovládání, které by i tak byly intuitivní. Ačkoliv návrhy byly do jisté míry zajímavé, pro mobilní rozvržení se příliš nehodily a to především z důvodu velikosti displeje mobilních zařízení. Tyto návrhy byly vhodné spíše pro větší zařízení, konkrétněji tablet. Z těchto papírových konceptů se do finální aplikace dostaly pouze některé konvenční části (které byly později graficky přizpůsobeny *Material Designu*).

Po prvních neúspěšných pokusech jsem došel k závěru, že nejvhodnější bude se zaměřit na konvenci *Material Designu*, ze kterého vzešla větší část finální podoby aplikace. Ještě před realizací tohoto návrhu jsem vytvořil prototypový návrh, který měl průběžně ověřit papírové hodnoty získané z dotazníků. Jednalo se tedy o jakési pracovní prostředí, které nebylo rozvržené za pomocí designového dokumentu (*Material Design*). Díky tomuto návrhu jsem také získal lepší představu o tom, jak by mohla vypadat konečná podoba aplikace.

### 3.1.2 Ovládání obrazovek a barevná paleta

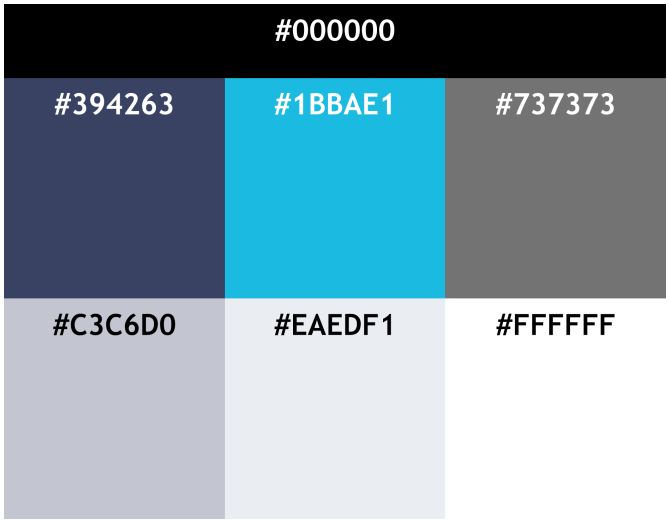
Při rozvrhování dostupnosti aktivit jsem si dopomáhal pomocnou tabulkou, ve které jsem měl načrtnuté, jaké možnosti by jednotlivé aktivity měly obsahovat.

Z tabulky 3.1 lze vyčíst, že Profil, Souhrn předmětů a Přehled týmů jsou globálními aktivitami aplikace, tudíž se na tyto obrazovky lze dostat odkudkoliv. Přirozeným ovládáním aplikace se uživatel nezanoří hluboko do dalších podaktivit a tudíž by neměla nastat nechtěná situace s hlubokým zanořením. Pokud by se uživatel nacházel hierarchicky na dně aplikace, musel by mnohokrát jít zpět, aby se dostal na požadovanou aktivitu, která by se vyskytovala hierarchicky na vrcholu. Rozvržení jsem vytvářel na základě aktuálního kontextu na obrazovce např.: Pokud se uživatel nachází na přehledu nástěnek (uskupené podle týmů), má na této aktivitě možnost vytvořit jak nástěnku, tak i tým. Naopak v přehledu týmů má pouze možnost vytvořit tým a editovat existující týmy.

| | **Profil** | **Souhrn předmětů** | **Přehled týmů** | **Vytvořit nástěnku** | **Vytvořit tým** | **Vytvořit předmět** | **Přesunout předmět** | **Editace nástěnky** | **Editace Týmu** | **Editace předmětu** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Přehled nástěnek** | X | X | X | X | X | | X | X | | |
| **Přehled týmů** | X | X | X | X | X | | | | X | |
| **Nástěnka** | X | X | X | | | X | X | X | | X |
| **Souhrn předmětů** | X | X | X | | | | X | | | X |
| **Profil** | X | X | X | | | | | | | |

Tabulka 3.1: Matice zobrazující které funkce by měly být dostupné z kterých obrazovek

Barevná paleta měla zprvu odstíny oranžové a bílé. Tuto volbu jsem posléze zavrhnul a barvy značně přizpůsobil tomu, jak jsou použity i ve webové adaptaci. Finální barevnou paletou se tedy stala tmavě modrá, tyrkysová a odstíny šedi (viz obrázek 3.1). Tmavě modrá jak u webové verze tak i mobilní verze plní funkci panelů. U webu vzhledem k rozvržení je tato barva využita pro menu a jiné větší zvýrazněné plochy. U mobilní verze se tato barva hodila především jako podklad pro Aplikační panel. Tato barva je také použita pro pozadí během načítání, aby byla změna barev při dokončení nahrávání dat ze serveru dostatečně kontrastní a upozornila tak uživatele na to, že už nemusí nadále čekat. Světlé odstíny byly použity zejména pro pozadí obrazovek nebo větších elementů, jako je například Karta (*Card – Material Design*) předmětu. Sekundární (doplňkovou) barvou je tyrkysová a stejně jak u webové aplikace, tak i v mobilní verzi je použita pro zvýraznění určitých informací (na webu například názvy nástěnek, statistiky nástěnek, či webová navigace). U mobilní verze je barva využita jako doplňková a pro zvýraznění změn či upozornění (například *Floating Button*) pro přidávání, zvýraznění nových záznamů a indikace nových záznamů). Zbylé odstíny šedi jsou využité pro pozadí, ikony a texty. Vybrané odstíny textu jsou odvozené podle toho, jaké mají pozadí (Světlejší odstín textu na tmavší pozadí atp.).

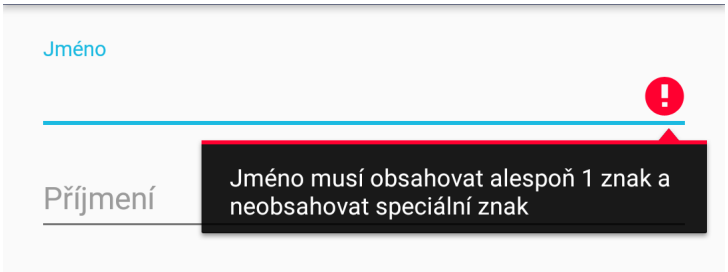



Obrázek 3.1: Použitá barevná paleta ve výsledné aplikaci

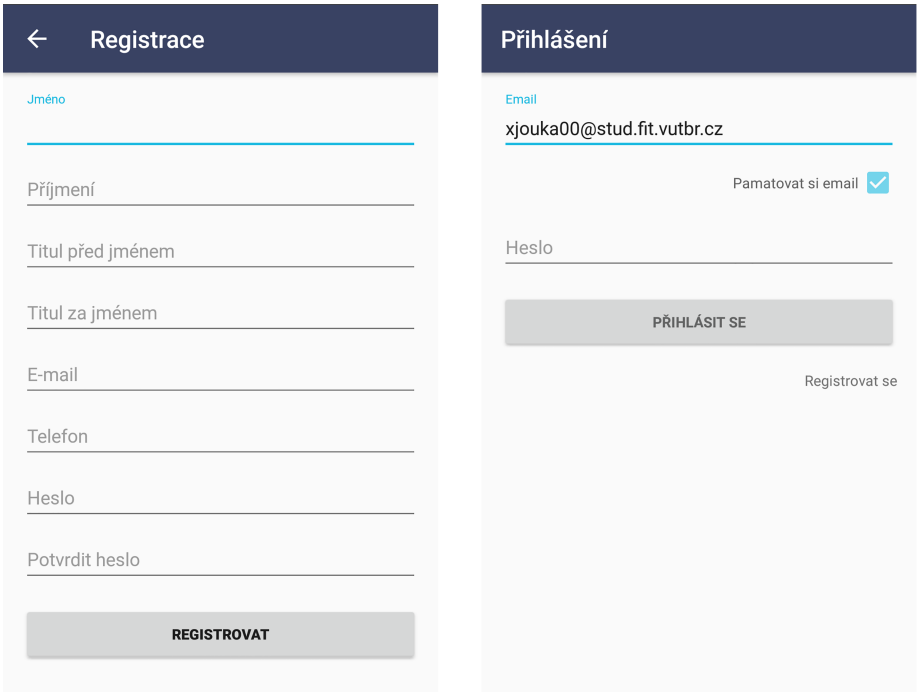
## 3.2 Přihlášení

Aplikace má uzavřený systém. to znamená, že bez předchozí registrace a následného přihlášení nelze v aplikaci pracovat. Jako první obrazovka se tedy zobrazí přihlašovací formulář (viz obrázek 3.3), ze kterého je možné přesunout se k registraci. V případě že uživatel nemá ještě účet, může ho vytvořit přímo v tomto bodu aplikace. Následně je přesměrován zpět na přihlašovací obrazovku. Pokud má uživatel účet již vytvořený, může se do aplikace přihlásit, čímž spustí proces ověřování. Před přihlášením je možné uložení mailové adresy, pro opětovné přihlášení, pokud se během užívání aplikace uživatel rozhodne odhlásit. Tato drobná funkce, která v prvotním návrhu nebyla zakomponovaná, překvapivě při testování pomohla několika uživatelům zjednodušit a zrychlit proces přihlašování.

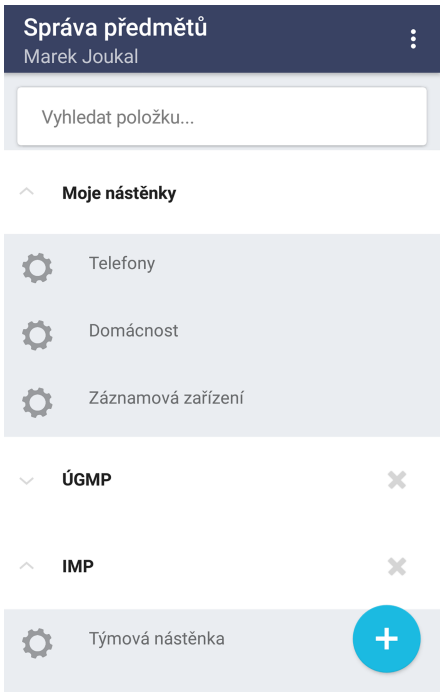
V rámci všech formulářů je použita implicitní metoda varovných upozornění pro případ, že uživatel vyplnil některý z formulářů chybně (viz obrázek 3.2). Zobrazují se, pokud uživatel nesplnil požadavky na heslo, jméno nebo jinou povinnou položku, která je nezbytná pro určitý formulář (například název předmětu při jeho vytváření).




Obrázek 3.2: Upozorňující zpráva při nesplnění některého kritéria při vyplňování formulářů




Obrázek 3.3: Obrazovka registrace do systému a přihlašovací obrazovka mobilní aplikace



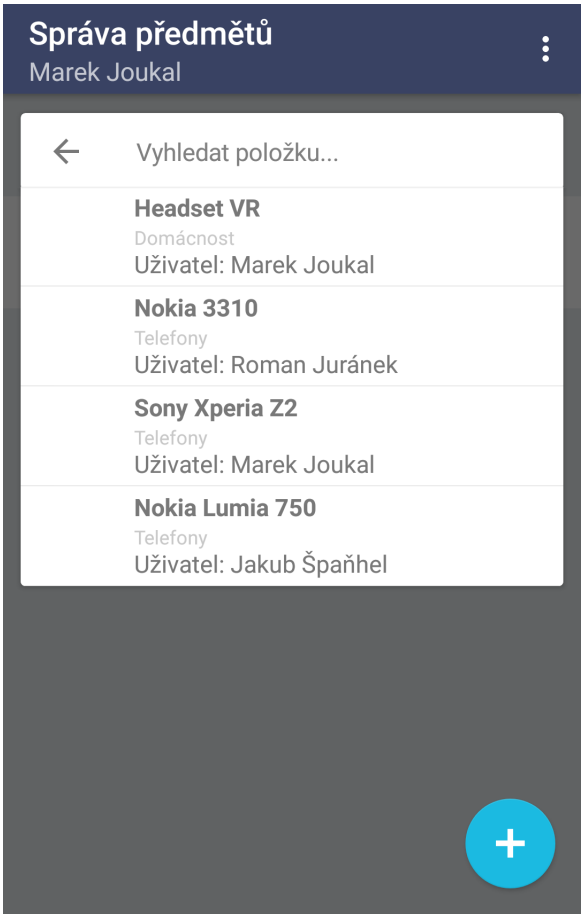

Obrázek 3.4: Obrazovka seznamu nástěnek. Nad seznamem se nachází vyhledávací panel a ve spodní části na pravé straně lze vidět *Floating Button*.

## 3.3 Seznam nástěnek

Po úspěšném přihlášení, nebo opětovném spuštění aplikace bez odhlášení, se uživatel ocitne na hlavní obrazovce – Seznamu nástěnek (viz obrázek 3.4). Na obrazovce je hlavním prvkem seznam týmů, ve kterých je členem. Tyto skupiny je možné rozkliknout, čímž se zobrazí nástěnky přiřazené právě tomuto týmu. Každá položka, kromě první, na úrovni týmu má také na pravé straně tlačítko pro možnost rychlého opuštění týmu. První položka tohoto seznamu jsou uživatelovy nástěnky (tudíž se nejedná u plnohodnotný tým), kam spadají všechny nástěnky, které vytvořil uživatel nebo do kterých je přiřazen jakožto explicitní člen.

### 3.3.1 Vyhledávání

Mezi aplikačním panelem a seznamem nástěnek se nachází vyhledávací panel. Pomáhá vyhledávat předměty podle jména, popisu nebo umístění v rámci všech nástěnek uživatele, ke kterým má přístup. Vyhledávací panel při zadávání názvu předmětu našeptává shodující se předměty (viz obrázek 3.5).




Obrázek 3.5: Vyhledávání v aplikaci zobrazí všechny shodující se položky s vyhledávaným výrazem. Pokud není zadán žádný řetězec, vyhledávání automaticky zobrazí všechny uživatelovy dostupné předměty napříč nástěnkami

Předměty se vypipisují hned pod vyhledávací panel. Vypsané položky obsahují krom svého názvu i nástěnku pro snadnější identifikaci předmětů s případně shodným názvem. Po kliknutí na hledanou položku se zobrazí dialogová nabídka čtyř základních operací – přesun předmětu, editace předmětu, výpis záznamů přesunu předmětu a smazání předmětu. Přesun předmětu má stejné ovládání jako na jiných místech aplikace, včetně možnosti změny umístění při přesunu. Záznamy předmětu mají analogické chování jako v seznamu nástěnky (viz kapitola 3.4.3). Dialogové okno je navrženo tak, aby se zavřelo pouze tehdy, kdy uživatel určitě dokončil svou interakci s předmětem. To znamená, pokud se uživatel rozhodne přesunout předmět a přesun nedokončí, nabídka zůstane otevřená. Stejně tak zůstane otevřená i po zobrazení záznamů či otevření editace předmětu. Pokud uživatel dokončí přesun, smazání předmětu, nebo klikne mimo dialogové okno, nabídka se zavře.

Vyhledávací panel je optimalizován pro rychlé ovládání. Proto je možné celý řetězec vyskytující se ve vyhledávacím panelu jedním klikem na příslušné tlačítko smazat. Pokud by chtěl uživatel pro stejný předmět otevřít nabídku vícekrát, po prvním vyhledávání se řetězec automaticky doplní na vyhledávanou položku. Dalším rozkliknutí panel automaticky začne vyhledávat podle doplněného řetězce.

### 3.3.2 Floating Button

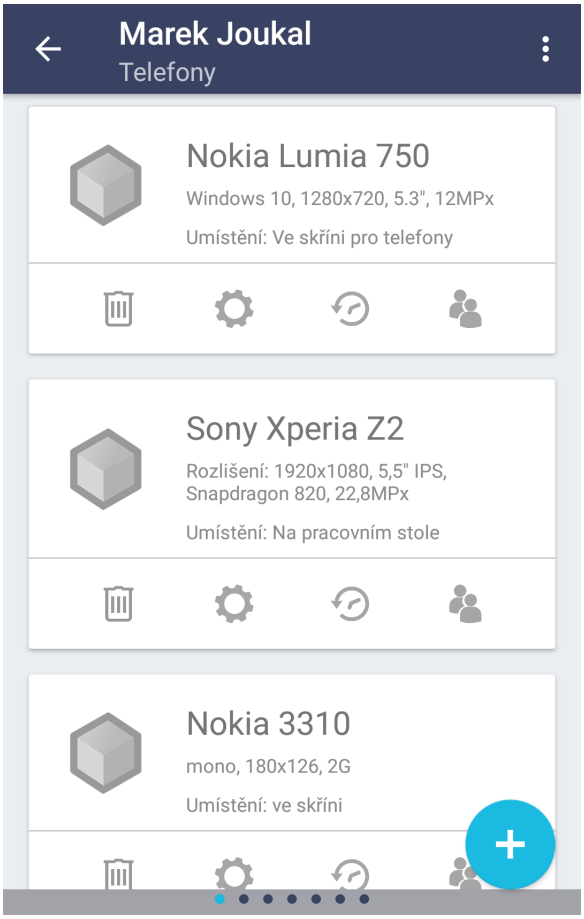
Na obrazovce s nástěnkami se vyskytuje také *Floating Button*[1] (viz obrázek 3.4) pro vytváření jak nových týmů, tak i nástěnek. Po stisku tlačítka se zobrazí dialogové okno pro výběr, co chce uživatel vytvořit. Jedná se o jedinou aktivitu, na které lze vytvořit nástěnku. Na konkrétní nástěnce *Floating Button* nenabízí vytváření nástěnky z důvodu nelogického kontextu a také proto, že na této pozici je tlačítko pro vytvoření předmětu. Ačkoliv zobrazení dialogového okna není standardním chováním, jedná se o unikátní případ, kdy tato volba dopomáhá intuitivnějšímu ovládání.

## 3.4 Nástěnka

Po klinutí na položku nástěnky v seznamu nástěnek je uživatel přesměrován na další aktivitu - Nástěnka (viz obrázek 3.6). Nástěnka zobrazuje informace o tom, kteří uživatelé mají k nástěnce přístup a kdo právě užívá nebo vlastní jaké předměty v nástěnce. Finální podoba je přizpůsobena *Material Designu – Cards* (dále jen karta). Toto rozvržení určuje, že jedna karta by měla popisovat právě jeden objekt a ne více. Forem karty je mnoho přičemž některé mohou obsahovat i obrázek. Zvolil jsem rozvržení s možností obrázku, protože podle něj lze předmět snadněji identifikovat. Možnost nahrání obrázku se v aplikaci nevyskytuje z důvodu aktuálně dostupných možností API serveru. Tato část je podrobněji rozebírána v kapitole 4.2, kde se zmiňuji o dostupných možnostech komunikace se serverem.

Pokud některý předmět obrázek nemá, na jeho pozici se objeví generický obrázek společný pro všechny předměty bez obrázku. Všechny elementy v rámci karty mají větší odsazení pro jednoduché dotykové ovládání, aby nedocházelo k překlepům. Ve spodní nabídce jsou redundantně umístěna tlačítka editace a přesunu předmětu. To je navrženo pro případ, že by pro uživatele byla intuitivnější možnost kliknout na tlačítko namísto dlouhého stisku pro přesun a kliknutí na předmět pro jeho editaci. Dále nabídka obsahuje možnost předmět smazat a zobrazit záznam přesunů předmětu. Na kartě se kromě názvu předmětu vypisuje popis předmětu (ten má omezenou délku v případě velmi dlouhého popisu) a jeho umístění.

[1] Prvek ve tvaru kruhu, který není zavislý na uspořádání zbytku obrazovky[3]




Obrázek 3.6: Otevřená nástěnka "Telefony". Lze vidět u uživatele tři předměty. Ve spodní části obrazovky se nachází Indikátor jednotlivých uživatelů a také *Floating Button* pro vytvoření nových předmětů.

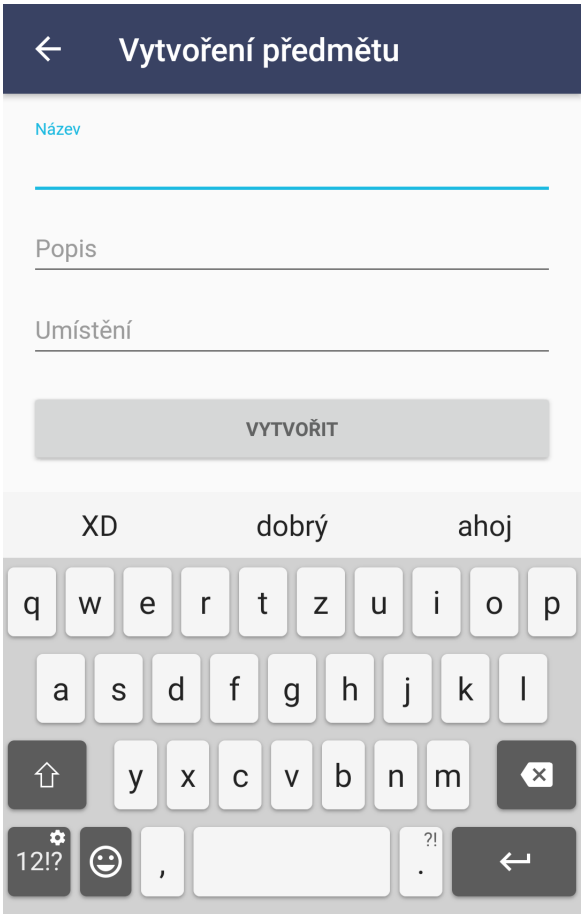
V aktivitě se opět ve spodní části obrazovky vyskytuje *Floating Button*, pomocí kterého lze přidat předměty do nástěnky. Předmět se automaticky nastaví do vlastnictví vytvářejícího uživatele, ačkoliv později ho lze změnit, pokud by došlo k případné změně vlastníka. Předmět lze přesunovat mezi libovolnými členy nástěnky, krom aktuálního uživatele přesouvaného předmětu. Tato možnost byla odebrána ve fázi testování z důvodu nepotřebnosti funkce.

Pokud se uživatel nachází v aktivitě nástěnky, v globálním menu se nachází možnost navíc vstoupit do nastavení nástěnky, ve které se právě nachází. Tato možnost se vyskytuje pouze v aktivitě nástěnky, protože v jiném případě není přesně určeno, kterou nástěnku uživatel zamýšlí editovat. Druhá možnost vstupu do nastavení nástěnky je v seznamu nástěnek v lokální nabídce nástěnky.

### 3.4.1 Vytvoření předmětu

Stěžejním prvkem pro nástěnky jsou mnohokrát zmíněné předměty. V tabulce 3.1 jsme mohli vidět, že předměty lze vytvářet pouze na obrazovce s vybranou nástěnkou. Předmět se díky tomu automaticky přiřadí k vybrané nástěnce. Povinným parametrem pro vytvoření

předmětu je pouze název. Další parametry jsou volitelné (viz obrázek 3.7). Těmi jsou popis a umístění pro podrobnější specifikaci předmětu. Popis může též sloužit k identifikaci, pokud by se v nástěnce vyskytoval jiný předmět se stejným názvem. Umístění informuje ostatní členy nástěnky, kde přesně se aktuálně předmět nachází, případně kde ho mohou najít, potřebují-li si předmět vypůjčit. Tento údaj lze měnit během přesouvání. Na konci formuláře pro vytvoření předmětu je umístěno tlačítko Vytvořit, kterým se uloží zadaná data a vytvoří se tak nový předmět.




Obrázek 3.7: Formulář pro vytváření předmětu. Obrazovky pro vytváření týmů a nástěnek se mu značně podobají.

### 3.4.2 Editace předmětu

Po vytvoření předmětu má uživatel možnost kdykoliv předměty upravit. Na aktivitu se lze dostat dvěma způsoby a to prostým kliknutím na kartu nebo kliknutím na příslušné tlačítko editace předmětu. Důvodem této redundance je možnost různých uživatelských návyků přístupu do určitých typů aktivit. Po vstupu do aktivity se automaticky předvyplní formuláře, které jsou identické s formulářem pro vytvoření předmětu. Aktivita se liší o jeden formulář navíc, který nastavuje vlastníka. Možnost změnit vlastníka je možná z důvodu, pokud by například původní vlastník měl odcházet z nástěnky, ale předmět má zůstat v nástěnce. Změnou vlastníka se předejde smazání předmětu při odstranění uživatele. Uživatel je tedy

schopný změnit vlastníka pokud například původní vlastník odchází ze zaměstnání a předmět měl pouze pracovně přiřazen. Na obrazovce editace je také možnost předmět kompletně smazat.

V původní verzi návrhu se v aktivitě vyskytovaly záznamy přesunů předmětu. Ty se původně měly vypisovat jako prostý text pod tlačítkem smazat. Po iteraci návrhu nástěnky se však tato část návrhu vypustila, protože by se jednalo o nekonzistentní logiku obrazovek. V editaci uživatel očekává aktivní prvky, se kterými může interagovat a měnit je, případně je mazat. Záznamy jsou přesným opakem. Nelze je jakkoliv měnit, a proto se přesunuly do samostatného dialogového okna. Pro uložení případných změn se v aplikačním panelu nachází tlačítko pro uložení. Na druhé straně aplikačního panelu se stejně jako u většiny obrazovek nachází tlačítko pro návrat zpět bez uložení změn. Po stisku tlačítka pro uložení je uživatel vyzván k potvrzení změn, pro případ nechtěného kliknutí na tlačítko. Uživatel je následně přesměrován zpět na obrazovku nástěnky s předměty.

### 3.4.3 Záznamy přesunů v nástěnce

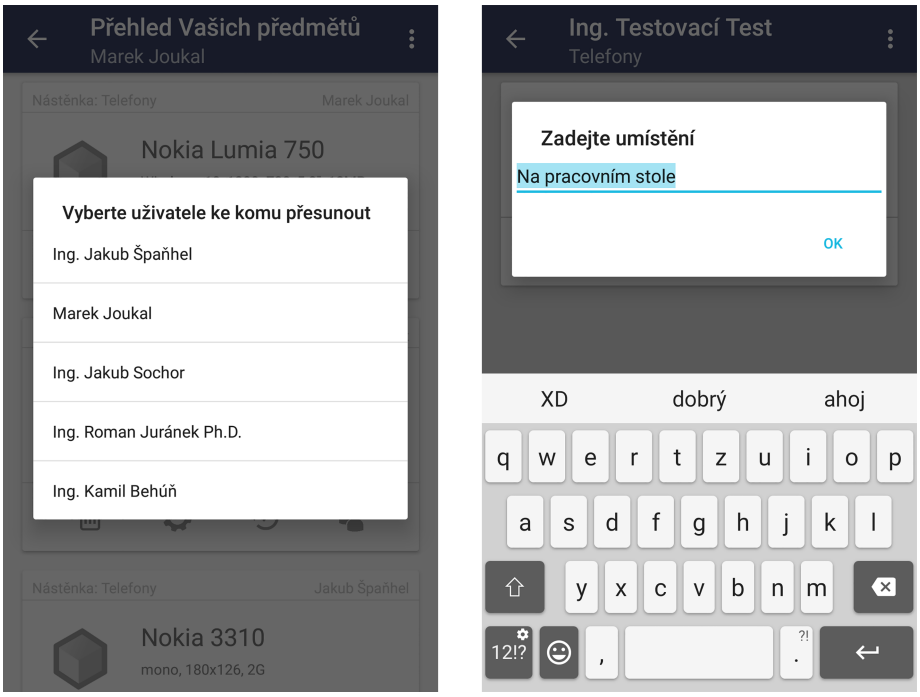
U každého předmětu se nachází jeho vlastní historie veškerých přesunů mezi uživateli. Přístup k záznamům z obrazovky nástěnky je mírně redukovanou podobou té, co se nachází na obrazovce souhrnu předmětů (viz obrázek 3.10). Záznamy předmětu se vypisují stejně, ale nezobrazuje se indikátor nových záznamů. Aplikace z důvodu přehlednosti indikuje pouze uživatelovy položky. Jelikož se nejedná o primární funkci, kterou uživatel vykonává v aplikaci, byla indikace pouze uživatelových položek lepší než, pokud by se zaznamenávaly položky všechny. Uživateli tím ovšem není zamezeno, sledování záznamů jiných předmětů, pouze o nich není informován malou indikační bublinou s číselným indikátorem počtu nových záznamů na daném předmětu od poslední kontroly.

### 3.4.4 Přesun předmětu

Jelikož přesun předmětu je nejčastěji využívanou funkcí, je možné ho provést z několika různých obrazovek. Jednou z nich je obrazovka nástěnky, kde je opět několik možností, jak uživatel může tento proces iniciovat. U každého předmětu se vyskytuje příslušné tlačítko pro přesun, ovšem přesunout předmět lze i pomocí delšího stisku na předmět.

V případě delšího stisku se nabízelo vícero možností, jaký typ interakce s uživatelem zvolit. Ačkoliv interakce typu Drag & Drop není nejrozšířenější v mobilních aplikací, jedná se o poměrně efektivní volbu, jak snadno může uživatel tento předmět přesunout. Na druhou stranu značnou nevýhodou tohoto typu přesunu je, že jakmile má k nástěnce přístup větší množství uživatelů, může nastat zdlouhavé přesouvání předmětu, než by se uživatel přesunul ke sloupci s adresovaným členem přesunu. Toto bylo důvodem mých úvah, zda tuto metodu má smysl skutečně implementovat, jelikož se jedná o velmi náročnou funkcionalitu po implementační stránce. Nakonec jsem usoudil, že optimálním řešením bude dialogové okno s uživateli (viz obrázek 3.8). Po vykonání akce pro přesun se tedy zobrazí dialogové okno, ve kterém uživatel určí, komu chce zvolený předmět přesunout. V seznamu nabízených uživatelů se nevyskytuje aktuální uživatel předmětu, z důvodu zbytečné operace. Existují dva způsoby, jak předmět novému uživateli přesunout. Prvním je jednoduché kliknutí na uživatele, kterému chce předmět přesunout. Druhým je delší stisk na uživatele, kde se na rozdíl od prvního způsobu zobrazí možnost změnit umístění. Tato možnost s nastavením umístění je úmyslně na variantě delšího stistku kvůli rychlosti základního přesouvání. Při návrhování aplikace jsem kladl důraz především na rychlost přesunu předmětu, ale také jsem chtěl uživateli umožnit změnit potřebné informace (například umístění). V případě, že

uživatel bude měnit umístění, automaticky se mu označí celý již existující text, díky čemuž ho nemusí zdlouhavě mazat. Celou akci lze kdykoliv zrušit kliknutím kamkoliv mimo dialogové okno, nebo pomocí systémového tlačítka zpět. Po dokončení přesunu se automaticky na pozadí vytvoří nový záznam o přesunu, který se následně odešle na server.




Obrázek 3.8: Seznam členů nástěnky, kterým lze přesunout předmět. Po delším stisku na jméno se objeví dialogové okno navíc s možností přepsat aktuální umístění předmětu.

### 3.4.5 Mazání předmětu

Funkce mazání nemá žádnou speciální funkcionalitu, je ovšem dostupná jak z editace tak přímo ze seznamu předmětů. Všechna čtyři tlačítka jsou rozvržena tak, aby uživateli pokud možno pomáhala intuitivně a rychle zvolit požadovanou akci. Například tlačítko pro mazání předmětu je záměrně umístěno jako první tlačítko zleva. Většina uživatelů drží svůj mobilní telefon v pravé ruce a mobil ovládají palcem, tudíž levá strana displeje telefonu je poměrně vzdálená. Je tak minimalizována situace nechtěné volby a zároveň se jedná o méně často používanou funkci v porovnání s ostatními funkcemi.

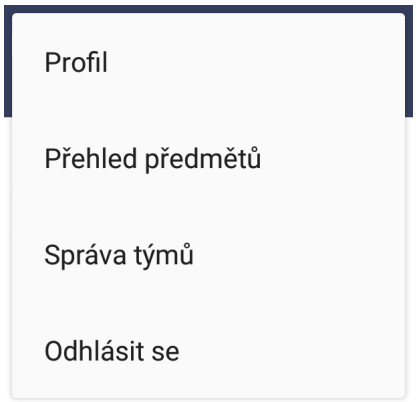
### 3.4.6 Indikátor fragmentů

Obrazovka nástěnky byla pro finální verzi navržena tak, že se v aktivitě vyskytuje vícero fragmentů, mezi kterými se lze přemisťovat pomocí gest pro posun doleva či doprava. Některé funkcionality, jako například právě přesouvání mezi uživateli za pomoci gest, mohou být obtížně zjistitelné. Gesta nejsou v aplikaci nijak znázorněna a tudíž je potřeba vytvořit prvky, které alespoň uživatele navede k vyzkoušení, zdali aplikace umožňuje sofistikovanější ovládání, případně funkce. Tyto prvky nejsou nezbytné u volitelných funkcionalit, které nejsou kritické pro využívání aplikace. Ovšem přesouvání se mezi jednotlivými uživateli

v nástěnce je příkladem nezbytné navigace. Aby byl uživatel seznámen s tím, že se v nástěnce vyskytuje více členů, a že se mezi nimi může přesouvat, nachází se ve spodní části obrazovky indikátor členů (viz obrázek 3.6) s vyznačenou aktuální pozicí.

## 3.5 Globální menu

Na všech obrazovkách, kromě přihlášení a registrace, se také v aplikačním panelu zobrazuje tlačítko pro zobrazení menu (viz obrázek 3.9). Z menu jsou dostupné globální aktivity, jimiž jsou dříve zmíněné Přehled předmětů, Profil a Správa týmů. K těmto aktivitám se tedy lze dostat odkudkoliv v celé aplikaci. Prvotní návrh obsahoval i volitelné zaškrtávací tlačítko, pro trvalé přihlášení, následně jsem však tuto volbu odstranil a udělal ji implicitní. K tomuto kroku mě vedla politika přihlašování mobilních aplikací, které se tak již chovají implicitně. Výchozí nastavení mobilních aplikací pracuje tak, že aplikace uživatele nechá trvale přihlášeného, dokud se sám explicitně neodhlásí. V menu tedy krom globálních aktivit zůstala pouze možnost odhlásit se. Ta přesměruje uživatele zpět na přihlašovací obrazovku. Globální aktivity se mohou v aplikaci otevřít pouze jednou, to znamená, že nemůže dojít k nekonečnému otevírání těchto obrazovek (a hypoteticky se zanořit nekonečně hluboko).



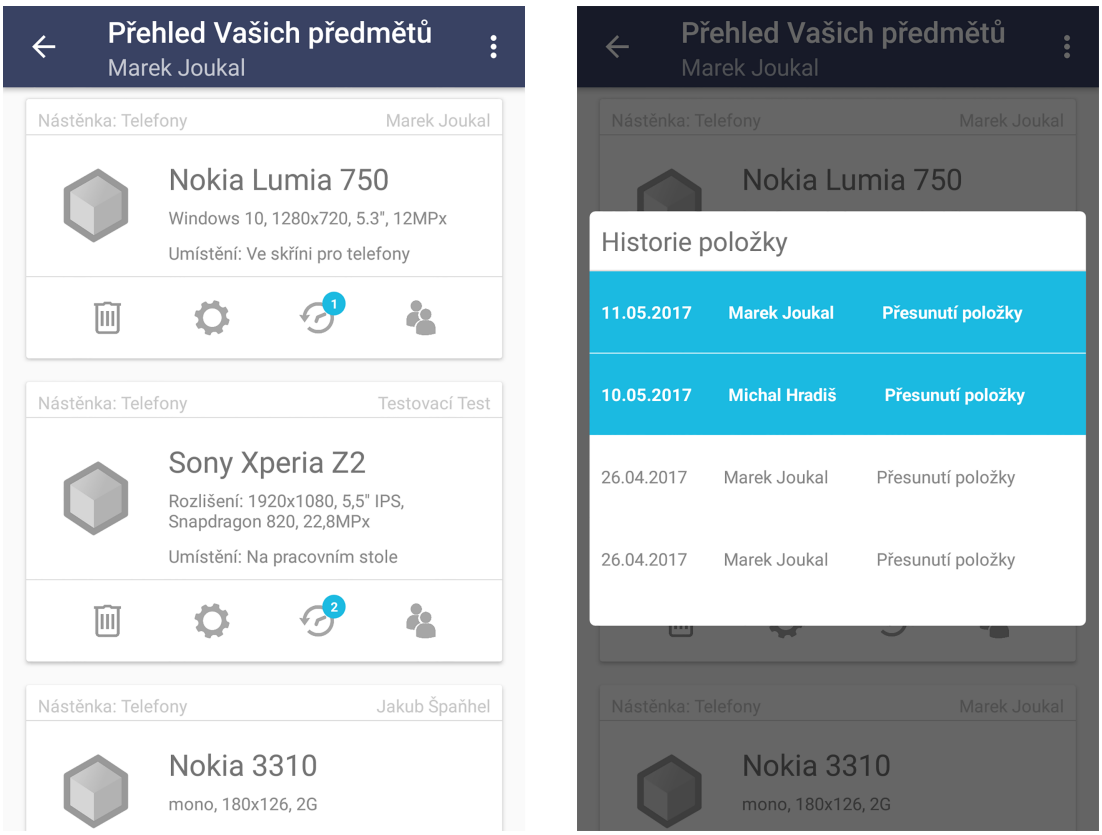

Obrázek 3.9: Globální menu, které je dostupné z naprosté většiny obrazovek aplikace. (výjimkou jsou editační obrazovky, ve kterých se na pozici menu nachází tlačítko pro uložení)

## 3.6 Přehled předmětů

Vedle hlavní obrazovky přehledu nástěnek se v aplikaci nachází podobně důležitá aktivita. Přehled předmětů je globální aktivita a tudíž k ní může uživatel přistoupit z kteréhokoliv místa aplikace. Aktivita má za úkol zobrazit souhrn všech uživatelových předmětů napříč všemi nástěnkami, ke kterým má přístup a vlastní v nich nějaký předmět. Přehled předmětů byl přidán a navrhnut až v pozdější fázi vývoje. Nejedná se tedy o součást prvotního návrhu. Aktivita byla inspirována věcnými připomínkami z dotazníku B, kde několik testujících uživatelů zmínilo, že jim občas schází přehledný souhrn všech předmětů, které vlastní napříč nástěnkami.

Předměty jsou uspořádány do jednoho seznamu, který se značně podobá nástěnce (viz obrázek 3.10). Narozdíl od nástěnky se zde ale nenachází fragmenty. Z důvodu konzistence vzhledu jsou položky opět vyobrazeny jako karty. Ty mají oproti standardní kartě navíc

dvě položky v záhlaví, jelikož předměty jsou sdruženy z různých nástěnek. První nalevo je název nástěnky, do které předmět spadá. Jelikož předměty nejsou uskupené do fragmentů (jako v nástěnkách) podle uživatelů, na pravé straně od názvu se nachází aktuální vlastník. Krom těchto dvou prvků, má karta předmětu oproti nástěnkové verzi o jednu funkcionalitu navíc. Aktivita u předmětů uživatele navíc zaznamenává změny výpůjček předmětu. Pokud je předmět přesunut jinému uživateli, automaticky se zaznamená tento pohyb a majiteli předmětu se následně ukáže či aktualizuje indikátor nového záznamu od jeho poslední kontroly záznamů. Po stisku příslušného tlačítka se zobrazí dialogové okno se záznamy a nové záznamy budou zvýrazněny.




Obrázek 3.10: Souhrn předmětů a záznamy o přesunech vybraného předmětu. Nové záznamy jsou v tabulce zvýrazněné

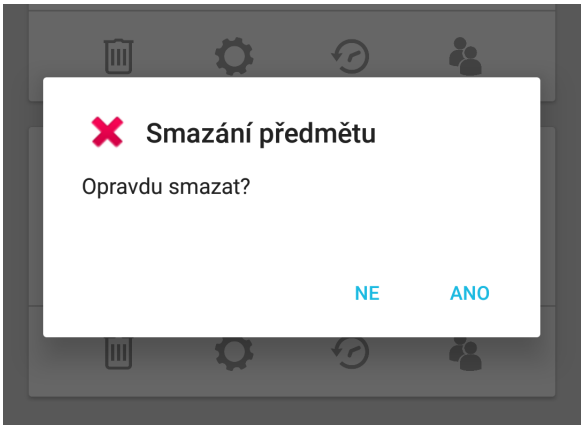
### 3.6.1 Správa týmů

Uživatelé mohou vytvořit libovolně velké nástěnky s neomezeným počtem předmětů, tak i členů. v případě firemního využití se může vyskytnout potřeba vytvořit více nástěnek, ovšem se stejnými uživateli. Vezmeme-li v úvahu, že i středně velká firma může zaměstnávat více jak sto zaměstnanců, nastává situace, kdy by správce musel každé nástěnce ručně přidávat individuálně každého zaměstnance. Aby se předešlo takové situaci, aplikace umožňuje zaměstnance, resp. uživatele, sdružovat do týmů. Uživatel může být členem více týmů současně. Týmy poté mají obdobné chování jako přidávání uživatelů do nástěnek. Pokud se přiřadí tým k nástěnce, všichni členové přiřazeného týmu získají přístup do nástěnky, po-

kud už ho k ní nemají. Tým může smazat pouze vlastník týmu, zatímco opustit tým může kterýkoliv uživatel. Pokud chce tým opustit vlastník, musí prvně zvolit nového majitele týmu. Aplikace tuto situaci automaticky detekuje a dosavadního vlastníka vyzve k vybrání nového majitele týmu. Možnosti přiřazování týmů, nástěnek a jejich opouštění je podrobně rozebráno v kapitole 4.6.

### Dialogová okna

Napříč celou aplikací je potřeba u uživatele ověřovat, zda chce zvolenou akci opravdu provést. Především se jedná o akce mazání položek, týmů či nástěnek, ale také editaci týmů, nástěnek, profilu a předmětu. Upozornění (viz obrázek 3.11) se také zobrazuje, pokud uživatel chce uložit vyplněný formulář pro přidání nového předmětu, týmu či nástěnky. Pokud jsou ovšem všechny formuláře prázdné, upozornění se neobjeví. Dialogová okna mají různou podobu grafických detailů, jako například obrázek křížku při mazání (viz obrázek 3.11). Pomocí dodržované konvence z jiných aplikací i platforem, je uživatel rychleji informován o podstatě dialogového okna.[9] Vyskakovací upozornění jsou navržená podle implicitního vzhledu používaného v systému Android.




Obrázek 3.11: Dialog upozorňující na smazání předmětu. Podobná okna doprovází celou aplikaci o provedencých změnách, rozpuštění týmů, atp.

### Kontextová zpráva

V aplikaci se nachází několik druhů dialogových oken, kde je nezbytná interakce uživatele. Krom toho je ale třeba informovat uživatele, že byla provedena nějaká akce, nebo případně informovat o vzniklé chybě během průběhu akce. Texty chybových zpráv nejsou podle pokynů *Material Designu* z důvodu snadnější identifikace chyby. Většina aktivit používá k identifikaci chyby svůj pracovní název a číselný kód, který dohromady dávají unikátní chybový kód. Platforma nabízí dva typy kontextové nabídky, resp. jejího vzhledu. První je implicitní, který je vždy stejný napříč všemi modifikacemi Androidu od různých výrobců mobilních telefonů. Druhý typ je stylizovaný podle uživatelova nastavení vzhledu v mobilu (viz obrázek 3.12). Jako lepší variantu jsem zvolil zprávy stylizované podle uživatelského nastavení v telefonu.


Obrázek 3.12: Kontextová zpráva aplikace při právě provedené události.

# Kapitola 4

# Implementace

V následující kapitole jsou popsané metody implementace aplikace. Rozebírají se zde způsoby a možnosti komunikace se serverem. Součástí je také popis způsobu přijímání a zpracování dat do objektové podoby a jak se s nimi pracuje jakožto objekty. v rámci implementace jsou také rozebrány použité externí knihovny.

## 4.1 Programovací jazyk a prostředí

Před zahájením vlastního vývoje aplikace bylo nutné rozhodnout v jakém programovacím jazyce bude nativní aplikace vyvíjena. Po zvážení pro i proti jsem se rozhodl pro jazyk Java. Ačkoliv byly k dispozici i další alternativy, v jazyce Java jsem měl již možnost programovat jiné projekty a tudíž mi základní syntaxe nebyla cizí. Xamarin[1] sice nabízí snadný převod vzhledu mezi jednotlivými platformami, ovšem aplikační logika je programovaná v jazyce C#, s nímž mám podstatně menší zkušenosti. Znamenalo by to podstatně více času potřebného pro pochopení syntaxe jazyka orientované pro vývoj mobilní aplikace. Podobná situace byla u dalších alternativ, kde se programuje programovacími jazyky, se kterými mám menší zkušenost než s jazykem Java. K vývoji aplikace jsem tedy nakonec zvolil Android Studio 2.3.1. Android Studio je vývojové prostředí vyvíjené v kolaboraci firmy Google a JetBrains. Prostředí je založené na IntelliJ IDEA a je kompletně zdarma. Jeho silnou stránkou je rychlost a snadné používání, kde veškeré potřebné aktualizace, opravné balíčky a jiné přídavné moduly mohou být automaticky staženy bez manuální obsluhy. Po programové stránce prostředí nabízí mnoho možností práce s kódem jako je našeptávání, navigace v kódu, refaktoring, analýza kódu, aj. Android Studio má v sobě integrovaný grafický návrhář, ve kterém je možné navrhovat rozvržení aktivit a to buď přímo v XML, nebo v tzv. Design módu. V obou případech prostředí zobrazuje náhled upravované obrazovky. Rozlišení tohoto náhledu lze snadno měnit, pro práci s různými polohami či velikostmi mobilních displejů a to včetně tabletů.

## 4.2 Komunikace se serverem

Stěžejní částí v implementaci je získávání dat, které se mají sdílet s ostatními uživateli a pokud možno, mají být kompatibilní s daty ve webové adaptaci. Domluvil jsem se s kolegou, který realizoval webovou část aplikace, že mi zpřístupní informace z databáze se stejnou strukturou tak, abychom mohli využívat stejných dat. Strukturu databáze zkonstruoval

[1] Vývojové prostředí pro mobilní aplikace

podle vlastních potřeb a následně jsem svou strukturu dat přizpůsobil, tak aby odpovídala té jeho. Z postupu vyplývá, že jsem pracoval především s daty, které mi byly dostupné ze serveru za pomoci *Application Programming Interface* (dále jen API).

API je pojem označující rozhraní, díky kterému lze externě operovat s daty, které jsou poskytované skrze toto rozhraní. Možnosti implementované aplikace jsou přímo závislé na serverové části a tudíž aplikace nemůže přistupovat k jiným datům na serveru, které nejsou dostupné skrze API. Rozhraní, které mi bylo poskytnuto, využívá čtyři typy HTTP dotazů, které je možné odeslat na server. Jimi jsou dotazy GET, POST, PUT a DELETE.

Dotaz GET (dále jen GET) umožňuje čtení dat resp. vypsání dotazovaných dat. GET je používán napříč celou aplikací, kde jsou potřeba vypsat aktuální informace ze serveru. Některé informace se stahují předem na pozadí, aby aplikace měla tato data k dispozici k případnému okamžitému zpracování a vypsání na obrazovku. Formát dotazu se posílá jako součást adresy, kde adresa se skládá z:

$$http://hosting/API/koncovy\_bod/id\_dat/$$

Dotaz POST (dále jen POST) umožňuje vytváření nových záznamů do databáze. POST je používán kdekoliv se vytváří nové uživatelské složky, jimiž jsou nástěnky, týmy, předměty, záznamy o pohybu předmětu nebo registrace uživatele. Data se na server neposílají jako součást adresy, ale pomocí stejnojmenné dotazovací metody POST.

Dotaz PUT (dále jen PUT) se stará o veškeré úpravy již existujících dat. PUT se v aplikaci využívá k úpravám existujících záznamů v databázi, jako je například přesouvání předmětu, změna popisu předmětu, změna názvu nástěnky či týmu atp. Data se opět odesílají jako soubor parametrů, obdobně jako POST.

Dotaz DELETE (dále jen DELETE) maže záznamy z databáze. Krom mazání nástěnek a týmů se DELETE využívá i pro mazání dat ve spojovacích tabulkách, záznamy udržující informaci o tom, který uživatel je ve kterých týmech či nástěnkách. Formát adresy posílaného dotazu je narozdíl od, POST a PUT, identický s GET, kde se na konec adresy vkládá identifikátor záznamu, který je požadován k odstranění.

K dispozici jsem měl přístup k základním tabulkám, jimiž jsou výpis všech uživatelů, výpis všech nástěnek, výpis všech týmů. Byla zde i možnost výpisu konkrétního týmu, uživatele či nástěnky podle identifikátoru, ovšem až na ojedinělý případ tento způsob přístupu nebyl optimální kvůli přístupové době a odpovědi ze serveru. Musel jsem proto využít dotaz na všechny záznamy z tabulky. Ačkoliv mi jsou tímto způsobem dostupná veškerá data, při větším objemu dat na serveru, tento způsob komunikace by nebyl v praxi použitelný, neboť by mobilní aplikace musela stahovat příliš velký objem dat. Krom velké zátěže na mobil, tento přístup vyžadoval větší objem dat, než by bylo potřeba. Tuto situaci by bylo vyřešit, pokud by API nabízela sofistikovanější přístupy k datům.

Po konzultaci s kolegou ohledně koncových bodů API mi byly poskytnuty některé optimalizované přístupy, které umožnili, že aplikace není tak zatížena stahováním velkého objemu dat. Tyto přístupy umožňili snadněji získávat všechny uživatele nástěnek na základě identifikátoru nástěnky. Druhým snadnějším přístupem bylo získání předmětů z nástěnky na základě identifikátoru nástěnky. Bohužel v případě uživatelů v nástěnce se do seznamu nevypisuje vlastník nástěnky, tudíž i tak je potřeba získat informace ze dvou různých míst. Navíc po získání identifikátoru majitele (přímo ze záznamu nástěnky), je potřeba ještě přiřadit identifikátor ke zbývajícím informacím o uživateli, jako je jméno, příjmení, tituly apod. Týmy žádnou podobnou možnost přístupu nemají, tudíž pokud je potřeba vypsat uživatelovy týmy, aplikace se musí dotázat na všechny týmy a poté vyfiltrovat seznam podle

identifikátoru uživatele.Jak již bylo zmíněno propojování, všechny vztahy mezi uživateli, nástěnkami a týmy jsou uložené ve třech spojovacích tabulkách:

- Uživatel-Nástěnka udržuje informaci, který uživatel má přístup ke kterým nástěnkám.
- Uživatel-Tým udržuje informaci, který uživatel je členem kterého týmu.
- Nástěnka-Tým udržuje informaci, který tým má přístup ke které nástěnce.

Zbývající tabulky jsou výpis veškerých záznamů předmětů a výpis záznamů na základě identifikátoru předmětu.

Data získávána ze serveru jsou v datovém formátu *JSON*, tzn., že všechny informace jsou posílány jako *JSON* objekt nebo pole (případně pole objektů). Přijímaná data napříč různými dotazy byly nekonzistentní jak z hlediska názvů tak i strukturálně. Určitá část klíčů byla pojmenována v angličtině a jiná v češtině. Některá data byla strukturovaná do pole objektů a některá byla místo pole vložena do jednoho velkého objektu. Navíc výsledná struktura se odvíjela od toho, zdali dotaz má nějaké výsledky či nikoliv. Pokud dotaz našel odpovídající výsledky, byly uskupené jako objekt, zatímco bez výsledku server odpovídal prázdným polem. Tyto nesrovnalosti značně komplikovaly zpracování odpovědí ze serveru, protože bylo potřeba ošetřit všechny tyto případy, aby se aplikace neočekávaně neukončila.

API neumožňuje nahrávání souborů a obrázků na server. Z tohoto důvodu tedy nelze v mobilní aplikaci k předmětům přidávat fotografie. Ovšem v rámci dalšího vývoje by se určitě jednalo o zajímavou funkci. Uživatel přidávající novou položku do nástěnky by mohl velmi snadno a efektivně přiřadit korespondující fotku, aby ostatní uživatelé mohli předmět snadněji identifikovat.

## 4.3 API Task

V aplikaci se nachází abstraktní třída API_task, jejíž implementace jsou využívané pro dotazy GET, POST, PUT a DELETE. v architektuře platformy Android není dovoleno provádět síťovou komunikaci v hlavním vlákně programu. Proto v Android SDK existuje speciální třída AsyncTask, která je nadstavbou pro práci s vlákny. s pomocí třídy AsyncTask lze snadněji implementovat kód prováděný mimo hlavní vlákno.[7]

Čtyři nejpodstatnější metody této třídy jsou:[7]

- `onPreExecute()` – metoda se volá jako první při spuštění asynchronní operace. Metoda se vykonává v hlavním vlákně, je tak možné přistupovat k prvkům GUI.
- `doInBackground()` – spustí se ihned po dokončení metody `onPreExecute()`. Metoda `doInBackground()` je prováděna ve vedlejším vlákně. Do této metody se vkládá veškerý kód pro vstupně-výstupní operace.
- `onProgressUpdate()` - metoda se provádí v hlavním vlákně a její invokace se provádí pomocí metody publishProgress(), která může být volaná z metody `doInBackground()`. Tímto způsobem je možno aktualizovat stav prováděné asynchronní operace.
- `onPostExecute()` – po dokončení metody `doInBackground()` je zavolána metoda `onPostExecute()`, která se provádí v hlavním vlákně. v metodě lze zpracovat výsledky operace, například zobrazit je.

V třídě API_task je implementovaná metoda `onPostExecute()`, která vrací výsledky stažených dat ze serveru ve formátu *JSON*. Každá ze tříd HTTP dotazu pak dědí od třídy API_task a implementuje vlastní metodu `doInBackground()`. Pro zpracování přijatých dat ze serveru jsem implementoval pro každý typ JSON objektu metodu, která transformuje JSON objekt na objekt třídy reprezentující nástěnku, předmět, uživatele, atp.

## 4.4 Struktura aplikace

Většina aplikace v podstatě zaobaluje, následně nějakým způsobem třídí a poté vyobrazuje seznamy dat, které získává ze serveru. Tyto seznamy jsou v implementační části modelovány jako objekty resp. seznam objektů. V aplikaci existují všechny typy objektů, které z API lze získat, tj. Nástěnka, Předmět, Tým, Záznam, Uživatel a jejich propojovací informace. Protože aplikace a její data jsou mezi sebou velmi provázaná, v aplikaci se vyskytuje globální třída navržená podle návrhového vzoru Jedináček[16]. Tato třída v sobě udržuje množství vlastností. Kvůli struktuře aktivit na platformě Android je do jisté míry obtížné přesunout některá data z jedné obrazovky do druhé. Pomocí metody Jedináček se stává přenášení nezbytně potřebných informací podstatně snadnější. Tento postup umožnuje mít dopředu relevantní data pro vyobrazení obrazovky, na kterou se uživatel může potencionálně přemístit. Globální třída udržuje seznamy uživatelů, předmětů a týmů, ale také v některých situací právě zvolenou nástěnku, právě zvolený předmět nebo právě zvolený tým. Tyto aktuální výběry se poté používají v jistých případech vyplívajících z kontextu (například když se uživatel přemístí na editaci předmětu). Mimo jiné, globální třída v sobě uchovává i řetězce pro komunikaci s API, v případě že by se adresa měnila.

Jak již bylo zmíněno, aplikace udržuje spojovací informace jako objekt obsahující dva identifikátory odpovídající danému uživateli, nástěnce či týmu. Aby bylo možné z těchto identifikátorů získat kompletní objekt, vytvořil jsem pomocné metody pro vyhledávání na základě identifikátoru. Protože tato spojení se využívají napříč celou aplikací, bylo logické tyto metody implementovat taktéž do globální třídy, zvlášť vzhledem k faktu, že je operováno s daty uloženými v této třídě.

## 4.5 Vyhledávání

O vyhledávání předmětů se stará vyhledávací panel na hlavní obrazovce. Pro základ vyhledávacího panelu je využita externí knihovna Floating Search View. Ta umožňuje z kteréhokoliv typu objektu vytvořit objekt kompatibilní s prvky (objekty), mezi kterými pak panel vyhledává. Požadovaný typ objektu, který má být vyhledáván, musí dědit od třídy *SearchSuggestion* a naimplementovat její metody. Těmi podstatnými metodami jsou `getBody()`, která vrací text, zobrazovaný ve vyhledávání a `describeContents()`, pomocí které lze přenášet identifikátor vyhledávané položky. Seznam zvolených objektů je posléze nutné předat panelu v aktivitě. Tento seznam se při kliknutí na vyhledávací panel ihned využije a vypíše položky, jejichž část se shoduje s vyhledávaným řetězcem. Vzhledem k tomu, že při prvním stisku je panel prázdný, vypíší se veškeré dostupné položky. Při každém napsaném či smazaném znaku se spustí událost detekující změnu v panelu. Touto událostí se průběžně aktualizuje vypsaný seznam shodujících se položek s aktuálním řetězcem v panelu.[2]

V panelu je možno vyhledávat jedním ze tří kritérií. Těmi jsou název předmětu, popis předmětu nebo umístění. Tato kritéria ovšem nelze kombinovat mezi sebou. Ačkoliv by existovala metoda, jak tuto možnost implementovat, k její realizaci jsem nakonec nepřistou-

pil, protože by mohlo dojít k zahlcení vyhledávání. v situaci, kdy uživatel má přiměřené množství dostupných předmětů ze všech nástěnek, by tato situace s největší pravděpodobností nenastala. Pokud by ovšem uživatel byl členem většího množství nástěnek s početným obsahem předmětů, mohlo by snadno dojít k uživatelsky nepříjemné situaci. Přestože by uživatel zadal poměrně specifické informace o předmětu, aplikace by mohla vypsat mnoho dalších předmětů, protože by mohlo dojít ke shodám mezi jednotlivými kritérii. Navíc by mohlo docházet k matoucím výsledkům, protože ve výpisu se nezobrazuje popis a umístění předmětu. Tyto textové prvky jsou z výpisu vypuštěny, aby jednotlivé položky nezabíraly na obrazovce příliš mnoho místa.

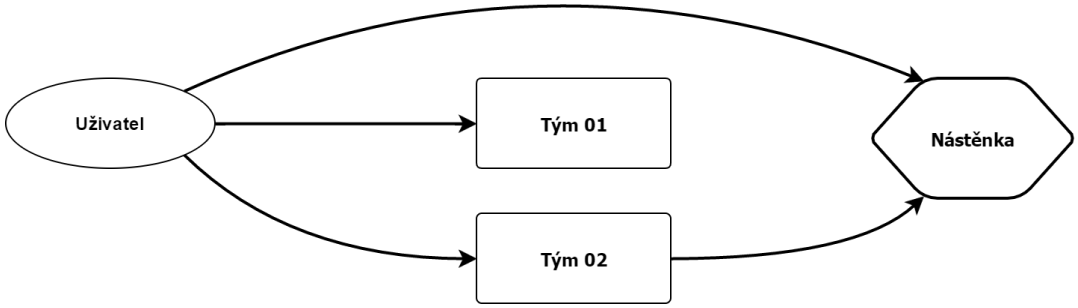
Jak již bylo zmíněno, tělo vyhledávacího prvku lze naformátovat pomocí návratové hodnoty metody `getBody()`. Ta může vracet formátovaný řetězec pomocí značkovacího jazyka *HTML*. Sada je ale oproti webové verzi omezená o některé elementy a naopak obohacena o atributy, pomocí kterých lze text nastylovat. K přestylování se používá třída Html z Android SDK.

Ve vyhledávání bylo potřeba upravit chování mobilního tlačítka zpět. Ačkoliv ve většině případů se tlačítko chovalo správně, ve specifické situaci tlačítko ukončovalo celou aplikaci namísto minimalizování vyhledávače. Situace mohla nastat, pokud uživatel vyhledal větší množství položek, které se nevlešly na obrazovku. Pokud uživatel začal listovat v seznamu vyhledávaných položek, a následně zvolil systémové tlačítko zpět, aplikace se minimalizovala. Abych tomuto jevu mohl zabránit, bylo potřeba tuto situaci detekovat. Naneštěstí byla tato situace velmi specifická a většina detekčních metod neumožňovala tuto situaci zachytit. Musel jsem tedy vytvořit pomocnou proměnnou, pomocí které jsem tyto změny ručně detekoval. Proměnná byla následně využita v metodě `onBackPressed()` přepisující původní chování, tak aby neminimalizovala celou aplikaci, ale pouze vyhledávací panel. I přes tento nedostatek externí knihovna usnadnila mnoho jiných částí implementace.

## 4.6 Přidávání a odebírání uživatelů

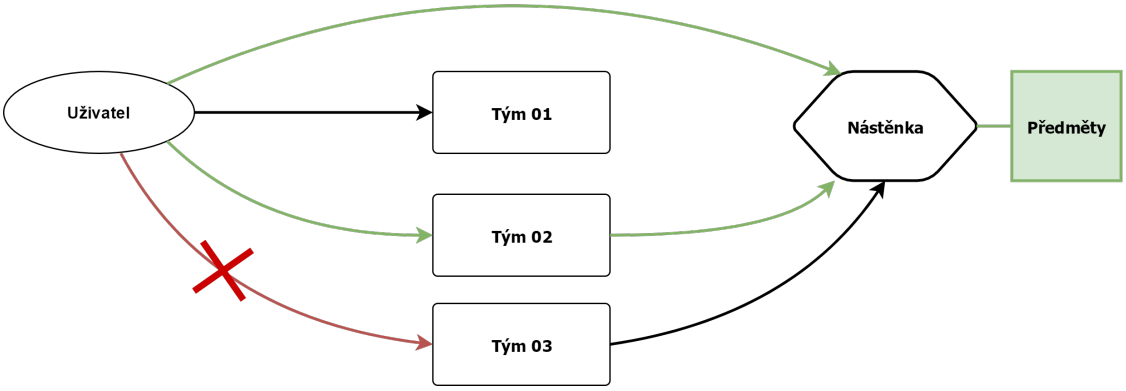
Ačkoliv editace či mazání nástěnek a týmů nejsou nejvyužívanějšími funkcemi aplikace, stále jsou nezbytné pro základní operace s aplikací. Aby mazání proběhlo úspěšně, je třeba v aplikaci zajistit odeslání příslušných požadavků na server. Při prvotní implementaci odebíracích funkcí jsem zjistil, že při smazání záznamu udržujícího identifikátor (například tým), které je součástí primárního klíče ve spojovacích tabulkách, databáze tyto záznamy nesmaže. Databáze se tak dopouští porušování integrity databáze, která může vyústit v nekonzistenci databáze. Abych alespoň částečně zamezil takovému stavu, bylo potřeba tuto integritu implementovat v klientské části. S tímto požadavkem ovšem přišlo i složité několika úrovňové cyklické procházení množin uživatelů, týmů, nástěnek, a dalších tabulek. Hlavním úkolem algoritmu bylo určit, zda uživatel ztratil po změně veškeré přístupy k nástěnce. V závislosti na předchozím stanovisku se algoritmus musí postarat o to, aby uživatel nepřišel o přístup do nástěnky, ale především, aby se nesmazaly jeho předměty v nesprávné situaci. Vzhledem k tomu, že má uživatel přístup z několika různých zdrojů, nastává situace, kdy je potřeba cyklicky projít všechny tyto možnosti.

Na obrázku 4.1 je znázorněný základní princip možností přístupů k nástěnce. Server uchovává informace o přiřazených uživatelích k nástěnce ve spojovací tabulce Board-User. Tak stejně se uchovávají informace o přiřazených členech k týmu v tabulce User-Team. Přiřazené týmy pak mají analogickou tabulku pro jejich přiřazení k nástěnce v tabulce Board-Team. Kromě dvou různých přístupů k nástěnce - jako explicitní uživatel (Board-User), nebo jako člen týmu (User-Team), může uživatel také k nástěnce přistupovat jako




Obrázek 4.1: Schéma jakými způsoby může být uživatel přiřazen k nástěnce. Skrze tým (tranzitivní závislost) nebo explicitně jako uživatel.

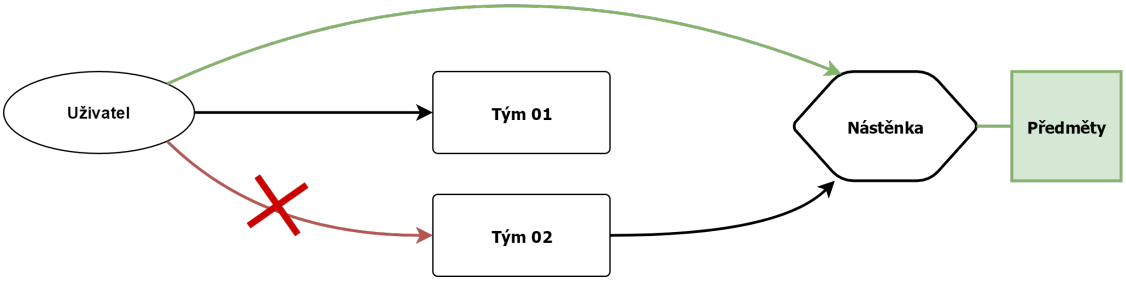
majitel nástěnky či majitel přiřazeného týmu. Majitelé nástěnky či týmu nejsou vedeni ve spojovacích tabulkách, nýbrž jako součást informací o nástěnce, resp. týmu (dále jen objekt). Kromě projití zmíněných spojovacích tabulek je tedy potřeba dodatečně zjistit, zda uživatel není majitelem jednoho z objektů. V případě mazání člena z týmu bylo potřeba složitější implementace, protože uživatel může mít vícenásobný přístup k nástěnce jako člen jiných týmů, které mají k nástěnce taktéž přístup. Při procházení přístupů k nástěnce bylo proto nutné zpětně zkontrolovat všechny k nástěnce přiřazené týmy, jestli uživatel není členem jiného týmu, který má přístup k nástěnce (příklad znázorněn na obrázku 4.2). V následujících příkladech jsou znázorněny situace, kdy uživatel není majitelem nástěnky ani týmu.




Obrázek 4.2: Uživatel opouští Tým 03. Avšak k nástěnce má stále přístup skrze Tým 02 i jako explicitní uživatel. Jeho předměty se nesmažou (a jím vypůjčené předměty se nepřemístí).

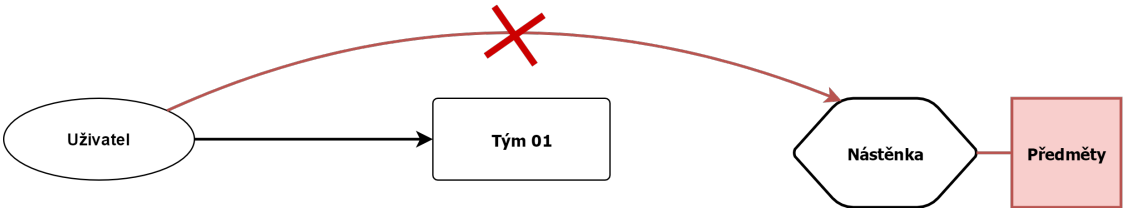
Pokud uživatel ztratí členství z týmu 03, musí se stejnou procedurou zkontrolovat všechny nástěnky, zda k nim nemá přístup z jiného zdroje. Na obrázku 4.2 má uživatel

přístup prostřednictvím týmu 02 a také sám jako explicitní uživatel, proto uživatelovy předměty zůstanou neměnné.




Obrázek 4.3: Uživatel opouští Tým 02. Avšak k nástěnce má stále přístup jako explicitní uživatel. Jeho předměty se nesmažou (a jím vypůjčené předměty se nepřemístí).

Na obrázku 4.3 uživatel ztratí přístup k týmu 02 tudíž i přes tranzitivní spojitost k nástěnce. Ovšem stále má přístup jako explicitní uživatel a předměty tedy zůstanou opět neměnné.




Obrázek 4.4: Uživatel opouští nástěnku skrze vlastní přístup. Po opuštění nástěnky uživatel ztratí svůj poslední přístup k nástěnce. Jeho předměty se smažou (a jím vypůjčené předměty se přemístí původním majitelům).

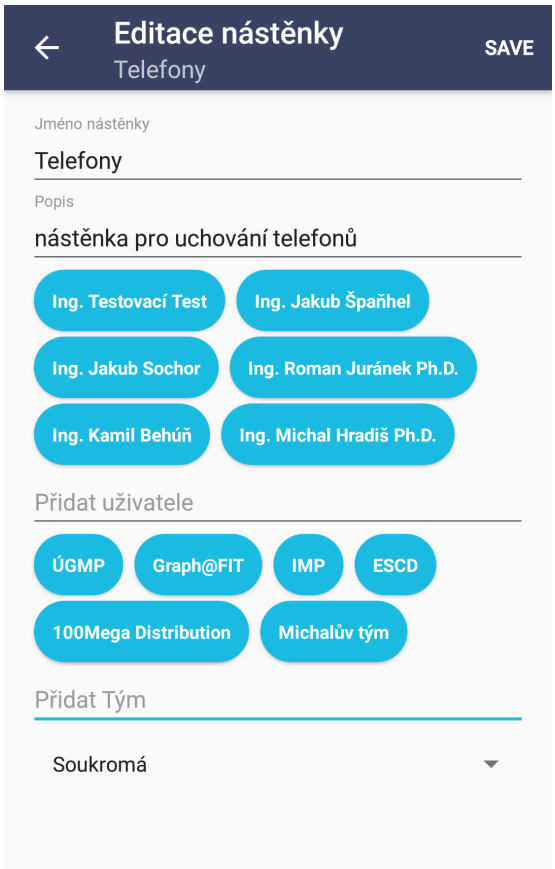
Na obrázku 4.4 uživatel ztratí svůj poslední přístup k nástěnce, algoritmus tedy vyhodnotí, že jeho předměty už nemají být k dispozici. Uživatelem vlastněné předměty se následně smažou a vypůjčené předměty jiných uživatelů, kteří přístup nadále mají, se vrátí původním majitelům.

Obdobný algoritmus byl potřeba vytvořit i pro editaci nástěnek. Hlavním rozdílem byla obměna kontrol kvůli kardinalitě vztahů. v nástěnce je potřeba při odebrání týmu všechny uživatele jednotlivě zkontrolovat, zda mají přístup k nástěnce z jiného zdroje.

### 4.6.1 Implementace vizuální stránky aktivity editace týmu a nástěnky

V editačních obrazovkách týmů a nástěnek je využito grafického prvku Chip (viz obrázek 4.5). Chipy například vyobrazují, kteří uživatelé či týmy jsou přiřazeni k nástěnce.[5] Aby se čipy správně zobrazovaly, bylo jim potřeba dynamicky nastavovat jejich šířku, podle délky jména. Následně se mohly uspořádat za sebou, bylo ale potřeba ošetřit přetékání chipů mimo obrazovku displeje. v opačném případě se chipy zarovnávaly do mřížky ne-

hledě na šířku jména. Proto jsem pro usnadnění implementace editačních obrazovek týmů a nástěnek použil externí knihovnu FlowLayout (kapitola 4.8.3). Externí knihovna přidává možnost rozvržení (Layout), které se stará o dynamickou šířku elementů uvnitř (chipů). Layout zároveň kontroluje, zdali je možno na řádek zobrazit další chip. Pokud ne, vloží se na nový řádek.




Obrázek 4.5: Chipy znázorňující uživatele a týmy přiřazené k nástěnce. U týmu si lze všimnout dynamického přizpůsobení v závislosti na délce názvu.

## 4.7 ViewPager

Téměř celá aplikace je rozdělena tak, že ke každé aktivitě je přidělena právě jedna třída, která dědí od třídy z Android SDK *AppCompatActivity*. Tím je řečeno, že se jedná o aktivitu, která zobrazuje celistvou část aplikace. Aktivita nástěnky ovšem využívá další funkcionality – Fragment.[8] Fragmenty jsou menší části, které se zobrazují v rámci jedné aktivity. Fragmenty lze využít na větších obrazovkách (tabletech), k zobrazení více informací současně. Pokud například uživatel otočí tablet na šířku a aplikace by byla optimalizovaná pro takovou zobrazovací plochu, aplikace zobrazí více fragmentů na jedné aktivitě. Není tedy třeba uživatele přemístit na zcela novou aktivitu. Vzhledem ke způsobu využívání aplikace, tato funkce nebyla primární.

Další důležitou funkcí fragmentů je jejich využití při implementaci ViewPager. ViewPager umožňuje vytvořit několik obsahů, mezi kterými se lze snadno přepínat pomocí gest. Fragmenty slouží jako nosiče těchto obsahů. V aplikaci je tento způsob navigace implementován pro jednotlivé uživatele v konkrétní nástěnce. Počet fragmentů je indikován pomocí Externí knihovny Material-ViewPagerIndicator, o kterém je psáno v kapitole 3.4.6.

## 4.8 Externí knihovny

Následující podkapitoly krátce shrnují jednotlivé externí knihovny, které byly v aplikaci použity.

### 4.8.1 Floating Search View

Floating Search View je externí knihovnou starající se o vyhledávání uživateli dostupných položek. Hlavní předností knihovny je velmi snadná implementace do zbytku aplikace. Pro začlenění do aplikace stačilo pouze nastavit položky, ze kterých se má vyhledávat a jak má vizuálně vypadat vyhledaná položka. Zbývající část implementace je volitelné přizpůsobení chování panelu. Dalšími výhodami knihovny je množství možností přizpůsobení chování panelu nebo grafické přizpůsobení.

### 4.8.2 Material-ViewPagerIndicator

Material-ViewPagerIndicator je externí knihovnou pro grafické vyobrazení počtu fragmentů v elementu ViewPager. Důvod použití této knihovny je jednoznačně triviální implementace. Element indikátoru stačí pouze vložit do XML souboru popisujícího rozvržení aktivity. Není tedy potřeba žádné programové obsluhy, ve které by programátor musel programově určit počet fragmentů. Všechny potřebné informace si element dohledá automaticky sám. Existuje zde i minimalistická programová alternativa, ta ovšem není pro implementaci povinná. Grafické nastavení se provádí přímo v XML souboru.

### 4.8.3 FlowLayout

FlowLayout se stará o korektní uspořádání chipů v editačních obrazovkách. Knihovna umožňuje implementovat layout, který dynamicky řadí své elementy a případně je vkládá na nový řádek. Layout usnadnil aplikování žádané obtékací logiky a nebylo potřeba přetékání a uspořádání elementů ručně ošetřovat. Ačkoliv existují další alternativy, zvolená knihovna FlowLayout byla pro potřebné účely naprosto dostačující.

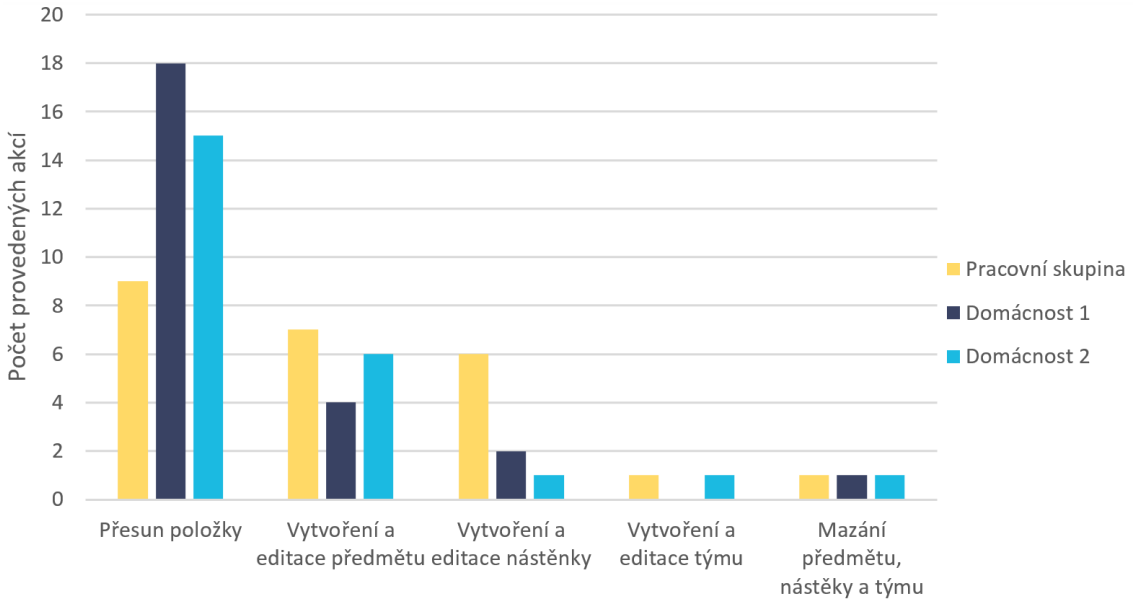
# Kapitola 5

# Testování

V následující kapitole jsou především rozebrány výsledky testování, které měly významný vliv na vývoj aplikace, a dopad zpětné vazby od testujících uživatelů.

## 5.1 Testování využívání funkcionalit

Testování bylo velice důležitou součástí vývoje programu. Díky němu jsem získával zpětnou vazbu a mohl jsem tak lépe určit, na které oblasti programu se mám více zaměřit. V počátečních fázích vývoje jsem potřeboval zjistit, jak často budou kromě hlavní operace (přesouvání předmětů) používané ostatní funkce aplikace. Tuto část jsem začal analyzovat již na prvním vývojářském prototypu aplikace na základě diskuze s testujícími. Ačkoliv testů proběhlo několik, neměl jsem možnost vyzkoušet aplikaci na větším množství uživatelů v měřítku organizace firmy. V případě takto vysokého počtu testujících, by se vyvozené závěry mohly lišit.



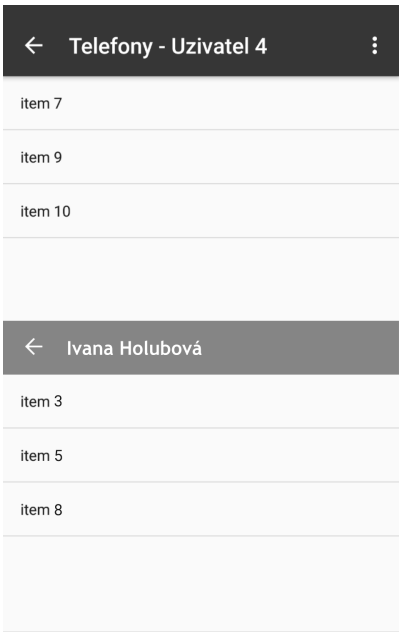

Obrázek 5.1: Graf výsledků prvního testování, které se zaměřilo na četnost využití jednotlivých funkcí aplikace.

První prototyp aplikace poskytoval především základní funkcionalitu a následně byl upravován dle výsledků testů. Po zprovoznění základních funkcí (zobrazení dat, přesouvání položek, vytváření předmětů, aj.) bylo učiněno několik testů ve skupině 6 lidí ve firemním prostředí a ve dvou domácnostech, aby bylo možné, vyvodit statistiky, které se alespoň vzdáleně podobají realitě. Pro účely testování jsem do aplikace implementoval vývojářskou konzoli s počítadly, která zaznamenávala počet provedených akcí. Výsledné hodnoty byly poté porovnány s daty získanými na základě interview (viz obrázek 5.1).

Ačkoliv byly všechny funkce využívány, dominantní funkcí stále zůstával přesun položek. Po prvotní analýze jsem dospěl k závěru, že bude vhodné soustředit se hlavně na dostupnost předmětů. Z testování také vyplynulo, že většina editací je prováděna hlavně ve fázi přípravy pro užívání. Vytváření týmů je především záležitostí větších skupin, než malých skupinek v řádu jednotlivců, tudíž se na grafu tyto operace téměř neprojevily. Skupina kolegů vytvořila několik různých nástěnek, zatímco domácnosti vytvořili jednu, maximálně dvě nástěnky. Tím se ukázalo, že aplikace může mít různé možnosti využití. Zatímco v práci se jedná především o pracovní předměty (kancelářské potřeby, nebo zařízení) v domácnosti se jedná o rodinné příslušenství.

## 5.2 Návrh aktivity nástěnka a jeho testování

Jednou z nejdůležitějších aktivit aplikace je obrazovka nástěnky. Zde proběhlo několik návrhů, jakou podobu by aktivita mohla mít. Přesněji řečeno bylo důležité určit správný způsob zobrazování předmětů a uživatelů, kteří mají do nástěnky přístup, tak aby bylo zobrazení přehledné a intuitivní. První návrh se pokoušel na obrazovce zobrazit dva seznamy uživatelů, kde u obou seznamů se po výběru zobrazí výpis položek vybraného člena (viz obrázek 5.2).




Obrázek 5.2: Původní návrh aktivity nástěnky. Obrazovka byla rozdělená na dvě části. Každou částí byl reprezentovaný jeden ze dvou uživatelů.

Po kratším testování přesunů předmětů mezi uživateli, jsem ovšem došel k závěru, že se jedná o velmi nepohodlný přístup, protože bylo potřeba provést více nadbytečných interakcí, než se podařilo přesunout jeden předmět mezi dvěma uživateli. Kromě problému s přesouváním, byl problém s tím, že uživatel neměl přehled, kdo aktuálně u sebe drží jaké předměty. Ačkoliv se mi ovládání zdálo nepřívětivé, než jsem začal vytvářet nový a intuitivnější návrh, vyzkoušel jsem reakci u několika testujících, abych získal zpětnou vazbu (na základě diskuze s testujícím uživatelem), kterou bych mohl uplatnit při novém návrhu. Z průzkumu se mi dostalo vpodstatě jednotné reakce. Hlavním nedostatkem byla, podle respondentů, nepříjemně obtížná obsluha.

V druhém návrhu jsem odstranil rozpůlení obrazovky, protože se jednalo o element, který svým způsobem znepřehledňoval zobrazené informace na obrazovce. Tím ale vznikla potřeba zobrazit ostatní uživatele, aby měl uživatel možnost přesouvat předměty mezi ostaními členy nástěnky (i ty předměty, které nemá právě u sebe). Dalším stanoveným cílem návrhu bylo zobrazit všechny předměty, které se v nástěnce vyskytují, resp. vytvořit přehledný seznam předmětů, ze kterého by uživatel jasně viděl kdo má jaké předměty. Pro návrh jsem využil dříve zmíněného ViewPager, který se stará o možnost zobrazování vícera fragmentů do horizontálního seznamu. Uživatel se přesouvá z jednoho fragmentu na další pomocí gest (doleva a doprava) v rámci celé obrazovky. Jednotlivý fragment, který repzentuje člena nástěnky, obsahuje seznam jeho aktuálně vlastněných předmětů. Po výběru předmětu se zobrazí jeho detaily, tak jak je možno vidět ve finální podobě, jen s tím rozdílem, že se v těle aktivity vyskytovalo ještě tlačítko pro uložení změn.

Po dalším testování jsem přistoupil k následující fázi vývoje aktivity a vylepšil veškeré její aspekty, tj. intuitivitu, přehledné rozvržení, přidání dalších informací o předmětu a nových funkcí předmětu. Ponechal jsem fragmentové rozvržení uživatelů, jelikož se tento návrh osvědčil. V aplikačním baru jsem změnil formát jména uživatele a aktuálně zobrazené nástěnky. Ten se vypisuje jako sekundární nadpis pod jménem uživatele, nikoliv za ním a to z toho důvodu, že v případě delšího jména uživatele nebo názvu nástěnky, by se oba názvy na jeden řádek nemuseli vejít (viz obrázek 5.3).


Obrázek 5.3: Rozdíl mezi aktuální verzí aplikačního panelu a vývojovou verzí. Lze si povšimnout, že ve vývojové verzi se jméno uživatele nevešlo vedle delšího názvu nástěnky.

Z návrhu jsem z důvodu konzistence odstranil možnost vytvoření předmětu z globálního menu, kde se tato položka zobrazovala pouze v případě, že uživatel byl na aktivitě nástěnky. Tuto funkci jsem přesunul do *Floating Button*, které využívám i pro vytváření týmů a nástěnek na jiných aktivitách.

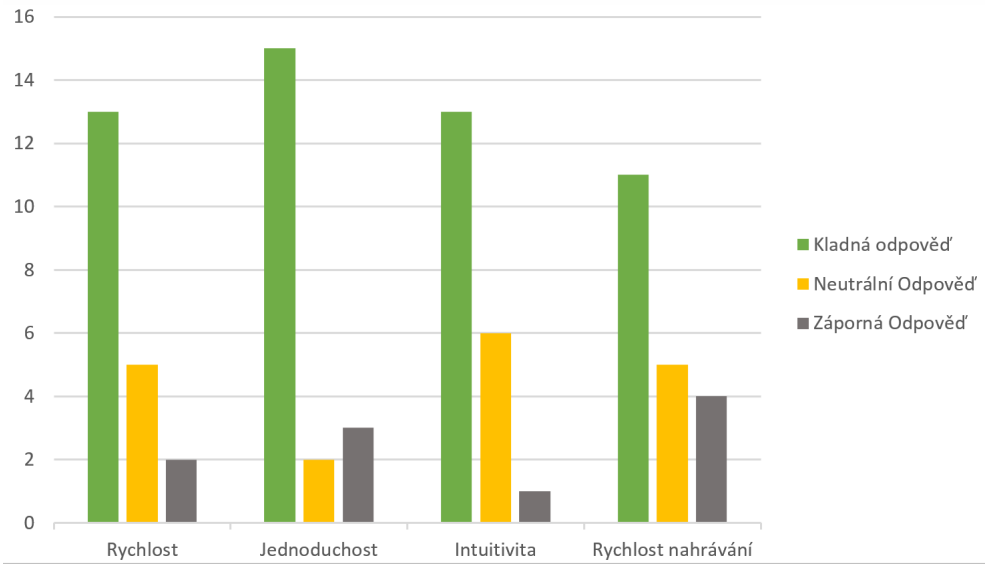
Hlavní změnou od předchozího návrhu je především vizuální změna a přizpůsobení podle designového dokumentu *Material Design*. Předměty uživatelů byly původně vypsané v prostém seznamu a po zvolení předmětu aplikace nasměrovala uživatele do detailů předmětu, kde se nacházel popis předmětu. Tak je tomu i u finální podoby. Delší stisk, stejně jako u finální podoby, ukázal dialogové okno pro výběr ke komu předmět přesunout.


Obrázek 5.4: Grafický prvek Cards (karta) stylizovaný podle pravidel *Material Designu*

Grafické rozhraní předmětu je navrženo podle stylového předpisu komponenty Cards[1]. Ikony ve spodní části karty jsou navržené podle četnosti užívanosti.

### 5.2.1 Finální testování

Po kompletizaci aplikace byl testujícím uživatelům předán seznam úkolů C. Uživatelé měli za úkol postupně provést všechny body obsažené v sadě. Pozoroval jsem jejich reakci na jednotlivé úkoly. Analyzoval jsem, zda se některý z uživatelů nezdržel na některé z úloh, a jakým způsobem úkoly realizovali. Po splnění všech úkolů dostali uživatelé krátký dotazník B pro zhodnocení intuitivity finální aplikace. Ve většině případů byla intuitivita hodnocena kladnou, případně neutrální volbou (viz obrázek 5.5).




Obrázek 5.5: Výsledky dotázníkuB.

[1]Prvek, který může obsahovat text, obrázek o individuálním předmětu. Může být použit jako kolekce různých objektů[4].

Konečná podoba mobilní aplikace dosáhla ve finálním testování uspokojivých výsledků, kdy nebyl neznamenán žádný dramatický výkyv negativní zpětné vazby. Po individuální diskuzi s každým uživatelem jsem se pro ověření zpětné vazby dotazoval na důvody jejich hodnocení. Valná většina se shodovala na tom, že aplikace je poměrně jednoduchá a není problém se v ní po krátké chvíli zorientovat. U některých prvků si zprvu nebyli zcela jistí, zda provádí chtěnou operaci. Nejčastěji bylo zmíněno opouštění týmu na hlavní obrazovce. Obrázek křížku na první pohled znejistil uživatele, zda se jedná o opuštění týmu nebo smazání týmu. Většina ovšem odvodila správný postup, protože při provádění sady úkolů si všimli ikonky popelnice na obrazovce se správou týmů.

Mírně vyšší zastoupení záporných odpovědí jsem oproti jiným kategoriím zaznamenal u rychlosti nahrávání dat (ze serveru). Tato situace nebyla natolik překvapující, protože serverová část není umístěna na výkonné dedikované serverové specifikaci. Mimo to rychlost stahování dat ovlivňuje do jisté míry způsob komunikace se serverem. Protože skrze API není možné filtrovat data, která jsou aktuálně potřeba, bývá poměrně často stáhován velký objem dat, aby informace byly co nejaktuálnější a zůstaly relevantní.

Během testování byly odhaleny drobné chyby, které byly následně odstraněny. Nejednalo se o žádné funkcionální chyby, ale z větší části pouze o špatné vykreslování. Testování proběhlo s uživateli, kteří se pohybují v prostředí informačních technologií i s průměrnými uživateli mobilních telefonů.

# Kapitola 6

# Závěr

Cílem této práce bylo navrhnout a implementovat mobilní aplikaci určenou pro širokou veřejnost, která umožňuje rychlou a efektivní správu položek. Aplikaci lze využít pro inventarizaci ve firmách, ale také pro snadnou organizaci vypůjčených předmětů v ústavech, školách, výzkumných centrech a jiných různých institucích. Systém umožňuje vytvářet týmy pro snadnější správu uživatelů. Uživatele tak lze organizovat do skupin, které je možno přiřadit k nástěnce.

Na základě testů se prokázalo splnění předem stanovených požadavků aplikace. Během testování se mi dostalo bohaté zpětné vazby, díky které byl vývoj aplikace snadnější. Diskuze s testujícími uživateli dopomohly ke změnám v aplikaci, které v původním návrhu nebyly zahrnuty. Na základě zpětné vazby jsem se tak mohl vyhnout nežádoucím komplikacím a zároveň si potvrdil předem stanovené cíle.

Aplikace by mohla být do budoucna rozšířena o zpětnou kontrolu vypůjčených předmětů při mazání člena z nástěnky. Mohlo by se jednat o kontrolu, zdali opouštějící uživatel nemá stále vypůjčené předměty. V případě, že by měl, aplikace by upozornila správce nástěnky na tuto skutečnost. Dalším rozšířením by mohla být možnost pořizovat a ukládat fotografii k jednotlivým předmětům. V obou případech by bylo pro realizaci zapotřebí rozšířit možnosti API serveru, díky kterým by šlo získávat filtrovaná data a ukládat fotografie na server. Budoucí rozšíření bude úzce souviset se zpětnou vazbou uživatelů a odrážet tak další vývoj aplikace.

# Literatura


[1] Český statistický úřad: *Chytré telefony zvyšují počet uživatelů internetu.* [Online; navštíveno 12.11.2016].
URL https://www.czso.cz/csu/czso/chytre-telefony-zvysuji-pocet-uzivatelu-internetu

[2] Cheshin, A.: *Floating Search View.* [Online; navštíveno 29.03.2017].
URL https://github.com/arimorty/floatingsearchview

[3] Design, M.: *Buttons: Floating Action Button - Components - Material design guidelines.* [Online; navštíveno 06.01.2017].
URL https://material.io/guidelines/components/buttons-floating-action-button.html

[4] Design, M.: *Cards - Components - Material design guidelines.* [Online; navštíveno 18.03.2017].
URL https://material.io/guidelines/components/cards.html

[5] Design, M.: *Chips - Material design guidelines.* [Online; navštíveno 21.03.2017].
URL https://material.io/guidelines/components/chips.html

[6] Design, M.: *Introduction - Material design - Material design guidelines.* [Online; navštíveno 11.11.2016].
URL https://material.io/guidelines/

[7] Developers, A.: *AsyncTask.* [Online; navštíveno 22.02.2017].
URL https://developer.android.com/reference/android/os/AsyncTask.html

[8] Horton, J.: *Android Programming for Beginners* . Packt Publishing, 2015, ISBN 978-1785883262.

[9] Konečný, M.: *Vyvíjíme pro Android: Dialogy a activity.* [Online; navštíveno 11.12.2016].
URL
https://www.zdrojak.cz/clanky/vyvijime-pro-android-dialogy-a-activity/

[10] StatCounter: *Desktop windows versions market share Worldwide | StatCounter Global Stats.* [Online; navštíveno 10.04.2017].
URL http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide#monthly-201603-201703

[11] StatCounter: *Mobile operating system market share Worldwide | StatCounter Global Stats.* [Online; navštíveno 10.04.2017].

URL http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201603-201703

[12] StatCounter: *Operating system market share Worldwide | StatCounter Global Stats*. [Online; navštíveno 10.04.2017].
URL http://gs.statcounter.com/os-market-share#monthly-201603-201703

[13] Svoboda, M.: *FAMU otevře katedru videoherního designu. Hry potřebují scenáristy, animátory i zvukaře, říká děkan*. [Online; navštíveno 10.02.2017].
URL https://magazin.aktualne.cz/kultura/film/famu-otevre-katedru-game-designu-studenti-diky-ni-najdou-pra/r~768cc6ccedde11e69aec0025900fea04/?redirected=1493401239

[14] Vokáč, L.: *80 % dospělé on-line populace má smartphone, nositelná elektronika na boom čeká*. [Online; navštíveno 12.11.2016].
URL http://tyinternety.cz/novinky/80-dospele-line-populace-ma-smartphone-nositelna-elektronika-boom-ceka/

[15] Web4trader: *Počet uživatelů chytrých mobilních telefonů se v ČR za pět let více než ztrojnásobil*. [Online; navštíveno 01.03.2017].
URL https://www.web4trader.cz/pocet-uzivatelu-chytrych-mobilnich-telefonu-se-v-cr-za-pet-let-vice-nez-ztrojnasobil-43829/

[16] Zbyněk Křivka, D. K.: *Principy programovacích jazyků a objektově orientovaného programování IPP - II*. [Online; navštíveno 28.02.2017].
URL http://pub.eyim.net/ziraficka/IPP-II-ESF-1_1_hyperlinks.pdf

# Přílohy

# Příloha A

# Průzkum trhu

1. Kolik mobilních telefonů vlastníte?
   - 1
   - 2
   - Více
2. Vlastníte tablet?
   - Ano
   - Ne
3. Jaký mobilní operační systém vlastníte (a primárně používáte)?
   - Android
   - iOS
   - Windows (Phone)
   - Symbian
   - BlackBerry OS
   - Jiný
4. Používáte nějaký systém TODO listů?
   - Ano
   - Ne
5. [Ano] Pokud ano, který?
   - Trello
   - Slack
   - Wunderlist
   - Asana
   - Todoist
   - Taiga
   - Jiné

6. [Ano] Jak často v něm operujete?
    - 1x měsíčně
    - 1x týdně
    - 1x denně
    - několikrát denně
7. [Ano] Kolik operací v něm průměrně uděláte?
    - 1 - 2 přesuny / splnění úkolů
    - 3 - 5 přesuny / splnění úkolů
    - Více než 5 přesunů
    - Spíše kontroluji, jestli se něco změnilo, než že bych měnil já
8. Představte si, že by váš nadřízený chtěl v práci udržovat informace o tom, kdo si co půjčil z firmy (případně z oddělení), dodržoval(a) byste administraci těchto přesunů? (to znamená zajistit v systému přesun této položky do vašeho inventáře)
    - Ano, nedělalo by mi to problém
    - Ano, ale dělalo by mi to problém
    - Ne
    - Nejsem si jist(á)
9. Zvedlo by vaše odhodlání, pokud byste mohl tyto přesuny udělat v rychlosti na mobilu?
    - Ano
    - Ne

# Příloha B

# Formulář pro ověření uživatelského rozhraní aplikace

1. Jak byste hodnotil(a) rychlost práce s aplikací? (jako ve škole)
   - (výborné, svižné) 1 2 3 4 5 (velmi špatné, zdlouhavé)
2. Jak byste hodnotil(a) orientaci v aplikaci? (jako ve škole)
   - (Jednoduchá, bez problému) 1 2 3 4 5 (Obtížné, neorientované)
3. Jak byste hodnotil(a) celkovou intuitivitu v aplikaci? (jako ve škole)
   - (výborné, intuitivní) 1 2 3 4 5 (velmi špatné, neintuitivní)
4. Jak byste hodnotil(a) rychlost načítání dat aplikace (jako ve škole)
   - (1) Výborné
   - (2) Až na výjimky, velmi dobré
   - (3) Průměrné
   - (4) Občas poměrně dlouhé
   - (5) Často dlouhé nahrávání dat
   - Poznámka:
5. 5. Jak byste popsal práci s aplikací, míru konkrétnosti nechám na vás. Pokud máte nějaké připomínky, co vás zpomalovalo při práci s aplikací, co jste naopak ocenil, nebo vám schází, zmiňte je:

# Příloha C

# Sada úkolů

1. Spusťte mobilní aplikaci pro správu a rezervaci předmětů
2. Přihlaste se pod uživatelem user_test (heslo: fit017user)
3. Seznamte se s uživatelským prostředím
4. Vytvořte nástěnku „Moje Nástěnka“ a přidejte člena Ing. Testovací Test a tým IMP
5. Zobrazte tuto nástěnku a prohlédněte zda, vidíte další členy, které jste přidal(a)
6. Vytvořte váš nový předmět s názvem „Předmět“ a zadejte umístění „doma“
7. Přesuňte předmět uživateli Ing. Testovací Test
8. Přesuňte předmět zpět k vám, ale při přesouvání změňte umístění na „venku“
9. Rozklikněte globální menu nabídku a přejděte do souhrnu předmětů
10. Podívejte se, jaké záznamy má váš předmět
11. Ze souhrnu předmětů přejděte do editace (vašeho) předmětu
12. Změňte libovolně popis a změny uložte
13. Smažte předmět
14. Vytvořte nový tým s názvem „tým [pořadové číslo]“ a přidejte do týmu uživatele Marek Joukal a Ing. Testovací Test
15. Přejděte do přehledu týmů a odeberte z týmu uživatele Marek Joukal
16. Vraťte se zpět na hlavní obrazovku a zkontrolujte, že tým vidíte v seznamu
17. Editujte vaší nástěnku a přidejte přístup vámi vytvořeného týmu
18. Podívejte se do nástěnky a zkontrolujte, jestli uživatel Marek Joukal je členem nástěnky.
19. Poté se pokuste opustit tým, který jste vytvořil(a). Vzápětí si to rozmyslete (po zobrazení dialogového okna) a tým namísto toho smažte
20. Opět se vraťte na hlavní obrazovku a zkontrolujte, zda tým zmizel ze seznamu

21. Přejděte do nástěnky a podívejte se, zdali uživatel Marek Joukal zmizel z nástěnky
22. Nástěnku smažte
23. Odhlaste se
24. Ukončete aplikaci